DECEMBER 1986

Year-End Special:
Free Annual Disk

DEC 9 1986

TechNotes

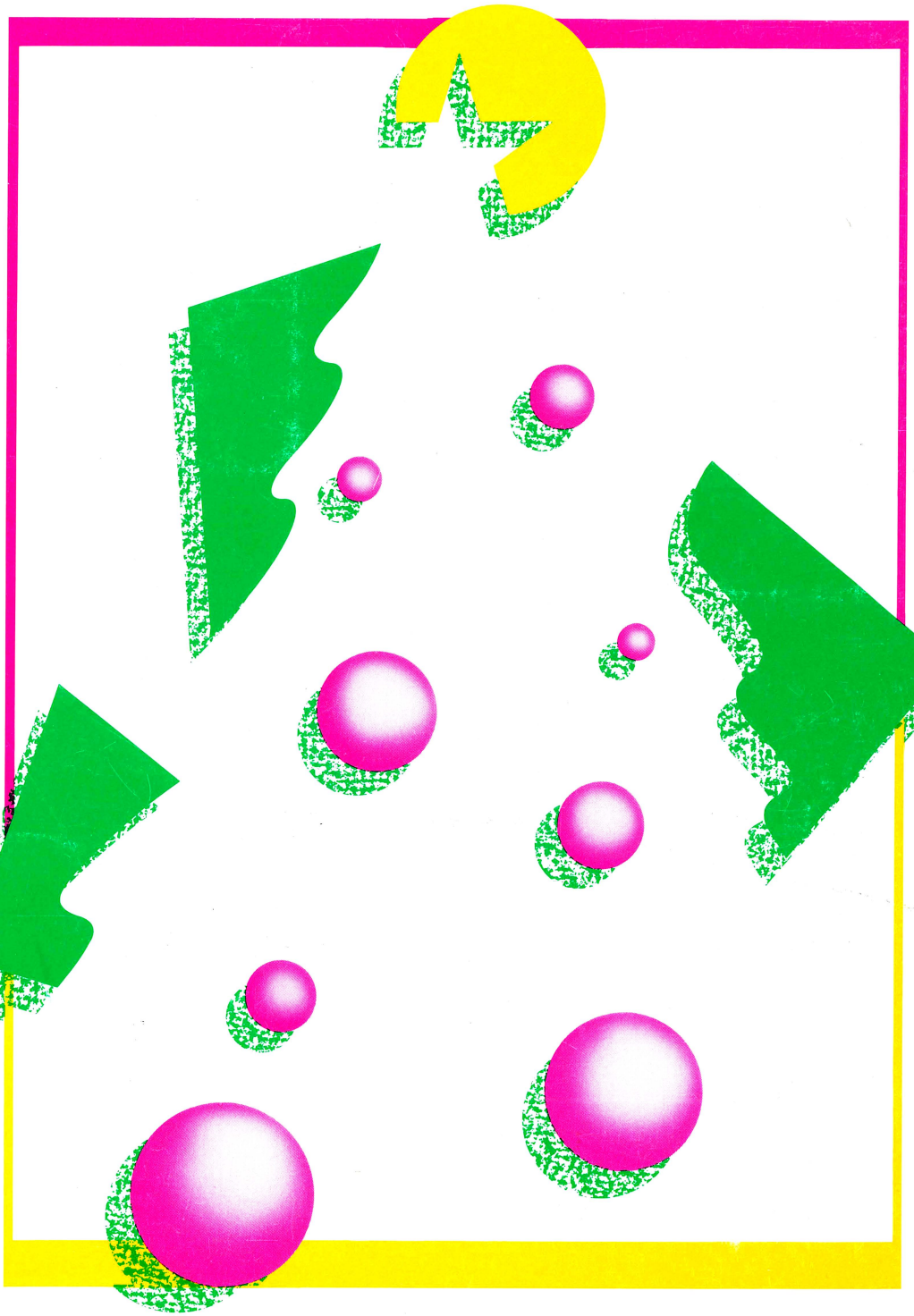
Creative STUFF()

Framework II Meets the Mouse

File Handle Status (dBASE III PLUS)

RBBS Dialer

1986 Annual Cumulative Index



ASHTON-TATE

AN ASHTON-TATE SOFTWARE SUPPORT PUBLICATION

TechNotes

TABLE OF CONTENTS

FEATURES

3 Creative Stuff
by J. Mike Hedblom and Christopher White

7 Framework II Meets the Mouse
by Kent Irwin

12 Change Summary for dBASE III PLUS Version 1.0—Part II
By Gail Kulikowsky

17 Sending Control Codes to the Printer
By Kenneth N. Getz, Larry Abare, and Karen Robinson

26 File Handle Status
by Kenneth N. Getz and Brenda Johnson-Grau

29 User-Defined Date and Character Fill Functions
by Jack Hirsch

32 RBBS Dialer
by Jack Hirsch

36 Amortization Schedule Using dBASE TOOLS FOR C
by Richard Bunter

41 REPORT FORM Templates in dBASE III and dBASE III PLUS
by Peter N. Gold

DEPARTMENTS

1 Read Me First
by Laurie Miller

45 Usage Tips

47 On Line Services

52 Multimate Technical Support

56 International

62 Letters

65 Corrigenda

71 Index

READ ME FIRST

by Laurie Miller

This issue marks the end of ***TechNotes***'s second year. From the introduction of our new, snappy design in the January issue to the inclusion of MultiMate and international technical material and excerpts from our on-line services, it's been a good year for ***TechNotes***. We also witnessed and reported on the release of dBASE III PLUS and the recent announcements of the Custom Support Plan and the removal of copy protection from Ashton-Tate products. Throughout, ***TechNotes*** brought you the latest and hottest technical information on Ashton-Tate and MultiMate software, and we will continue this tradition in the new year.

If you subscribed to ***TechNotes*** last year, you will recall that we included a disk in the December 1985 issue. Because of its huge success, we decided to do it again this year. The 1986 Annual ***TechNotes*** Disk contains the best application programs we published during the past year: random number generator for dBASE II; simulating arrays in dBASE II and dBASE III; searching, sorting, and reporting in dBASE III PLUS; assorted date routines for dBASE III; string functions in Framework II; sharing data with Framework II; and something we call the "Killer Library," a set of "quickie" but killer macros to help you automate your work in Framework II.

So that we may ensure safe delivery and provide you with the disk format of your choice, we have chosen not to mail the disk inside the magazine as we did last year. Instead, to obtain your 1986 Annual ***TechNotes*** Disk, please clip out the "Disk RSVP" (below), peel off and attach your mailing label, indicate which format you need, and mail it to us at:

expN2 :: = number of characters to replace

There are several rules to keep in mind when you use STUFF():

1. Negative numeric arguments are treated as zero.
2. There are no optional arguments.
3. There are no range errors.

TechNotes

Publisher
Ashton-Tate Software Support Center
Jay Hanson, Director
Roy Moore, Manager

Editor-in-Chief
Laurie Miller

Editors
Christopher White

Technical Writers
Brenda Johnson-Grau
Karen Robinson

Assistant Editor
Olivier Biggerstaff

Technical Staff
Larry Abare, Richard Augelli, Stanley Ballenger, Rodney Barbati, Jeffrey Bersin, Richard Bunter, Mike Carlisle, Ray Cortenbach, Fred Crownover, Michael Fiore, Rick Fillman, Charles Gehman, Ken Getz, Richard Goepel, Peter Gold, Mina Gussin, Doug Haman, Scott Haynes, Mike Hedblom, Clint Helton, Jack Hirsch, Errol Jackson, Matt Johnson,Rosaline Keenan, Gail Kulikowsky, Steven Kurasch, Jerry Levy, Chris Lo Vasco, Jeffrey McCrimon, Tom Mangelsdorf, Jerry Mendes, Vincent Mendillo, Elizabeth Miranda, Jeffrey Moore, Keith Mund, Joseph Perret, Jaspal Rattan, John Regen, Deborah Ridolfo, Daryl Saybold, Steve Silverwood, Joseph Stoltz, Naoma Stow, Ralph Wakefield, Lisa Wanek, Quinn Wildman, Roger Wegehoft,

Contributing Editors
Paul Bukauskas (International),

DISCLAIMER NOTICE

The information contained in this publication is designed to help you develop and improve your programming skills and to assist you in using Ashton-Tate products. Ashton-Tate assumes no responsibility whatsoever for the uses made of any of the software code in this issue, whether modified or unmodified.

If you discover what you believe is an anomaly (as defined under **Anomalies**) not presented in this issue or others, please send a detailed description of the possible anomaly to:

Ashton-Tate Software Support Center
20101 Hamilton Ave.
Torrance, CA 90502-1319

If the reported anomaly can be reproduced by our technicians, it may be added to a future issue of ***TechNotes***.

The policies that are described in ***TechNotes*** apply only to the United States (and possessions), Canada, and Mexico. Residents of other foreign countries should contact a local Ashton-Tate representative in their country for policy information and technical support.

Ashton-Tate and the authors will not be liable for special, incidental, consequential, indirect or other similar damages, even if we have been advised of the possibility of such damages. This means we are not responsible for damages or costs incurred as a result of loss of time, loss of data, loss of profits or revenues, or loss of use of the software. In addition, we are not responsible for damages or costs incurred in connection with obtaining substitute software, claims by others, inconvenience or similar costs.

No warranties, either expressed or implied, are made regarding the contents of this publication, its merchantability or fitness for any particular purpose, all of which are hereby expressly disclaimed. Note, however, that some states do not allow the exclusion or limitation of warranties or of direct, incidental or consequential damages, so these warranty limitations may not apply to you.

In no event will our liability for any damages to you or any other person ever exceed the price paid for your subscription to ***TechNotes***, regardless of any form of the claim.

Additional statements by agents, employees, distributors and dealers of Ashton-Tate do not constitute warranties by Ashton-Tate and do not bind

Creative STUFF()

by J. Mike Hedblom and Christopher White

The STUFF() function is a powerful and often overlooked tool for working with character strings. It can change part of a character string without disassembling it into pieces, manipulating the parts, and then reconstructing the new string. In addition to the potential brevity it can give your character string expressions, STUFF() is quite flexible allowing you to insert, delete, and replace portions of character strings with other character strings. Another advantage over other string manipulating functions especially the substring functions (SUBSTR(), LEFT(), and RIGHT()) is that STUFF() does return execution errors. The rules and capabilities are explained below.

How STUFF() Works

The STUFF() function returns a character value from four arguments. As stated on page 6-79 of *Using dBASE III PLUS*, STUFF() replaces a portion of a given character string with another character string based on two numeric arguments: the starting position of replacement and the number of characters to replace. The general syntax is as follows,

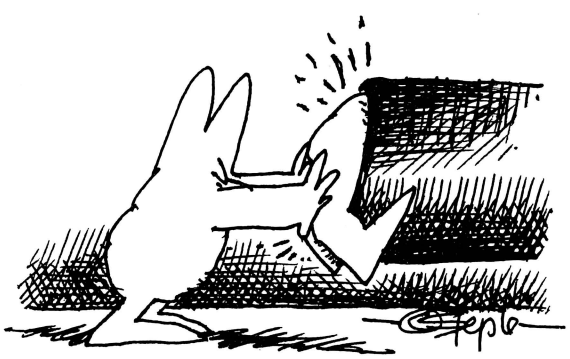
STUFF(< expC1 > , < expN1 > , < expN2 > , < expC2 >)

where

expC1 :: = string to replace into
expC2 :: = string to replace with
expN1 :: = starting position of replacement
expN2 :: = number of characters to replace

There are several rules to keep in mind when you use STUFF():

1. Negative numeric arguments are treated as zero.
2. There are no optional arguments.
3. There are no range errors.



General Capabilities

As stated above, STUFF() can perform a number of different operations on character strings that normally require a combination of substrings and concatenations if you use other dBASE III PLUS string functions. These general string operations include insertion, deletion, and replacement. In addition, STUFF() allows you to perform combinations of these general capabilities within one STUFF(). The following list details the possible configurations of arguments that give you insertion, deletion, and replacement.

1. Simple insertion

   If the second numeric argument is less than one, STUFF() inserts the replacement string into the target string at the position specified by the first numeric argument. For example,

   ```
   cstring = "1234567890"
   ? STUFF(cstring, 1, 0, "abcde")
   abcde1234567890          <————Result.
   ```

   Here, the string "abcde" is inserted at the first character position in the memory variable "cstring".

2. Simple replacement

   If the second numeric argument is greater than one and the same as the length of the replacement string, STUFF() replaces into the target string at the "starting position" (the first numeric argument) and replaces for the number of characters specified by the second numeric argument. For example,

   ```
   cstring = "1234567890"
   ? STUFF(cstring, 1, 5, "abcde")
   abcde67890               <————Result.
   ```

   Here, positions one through five of the memory variable "cstring" are replaced with the string "abcde". If the second numeric argument is greater than the length of the replacement string, you will get replacement with deletion as explained below.

3. Simple deletion

   Whenever the replacement string expression evaluates to a null (length of zero), STUFF() deletes the number of characters specified by the second numeric argument from the target string beginning at the position specified by the first numeric argument. For example,

   ```
   cstring = "1234567890"
   ? STUFF(cstring, 5, 1, "")
   123467890                <————Result.
   ```

   In this example, the character "5" located at the fifth position in "cstring" is deleted.

4. Replacement and insertion

   Whenever the length of the replacement string is greater than the "numbers of characters to replace" argument (the second numeric argument), STUFF() replaces from the "starting position" (the first numeric argument) up to the "number of characters to replace" and then inserts the rest of the replacement string. For example,

   ```
   cstring = "1234567890"
   ? STUFF(cstring, 1, 4, "abcde")
   abcde567890              <————Result.
   ```

In this example, "abcd" replaces the first four positions in "cstring" and then the character "e" is inserted at position five.

5. Replacement and deletion

   When you specify a "number of characters to replace" argument (the second numeric argument) longer than the replacement string, STUFF() replaces the target string with the replacement string and then deletes characters from the end of the replacement string to the position in the target string specified by the second numeric argument. For example,

```
cstring = "1234567890"
? STUFF(cstring, 1, 8, "abcde")
abcde90        <———Result.
```

   In the example above, STUFF() replaced the first five characters of "cstring" with "abcde" (which is five characters in length) and then deleted the next three characters.

6. Replacement and deletion of remaining

   If you specify the "number of characters to replace" (the second numeric argument) is longer than the target string, STUFF() replaces from the "starting position" (the first numeric argument) until there are no more characters in the replacement string and then it deletes the rest of the characters in the target string. For example,

```
cstring = "1234567890"
? STUFF(cstring, 2, 80, "abcde")
1abcde         <———Result.
```

   Here, STUFF() replaces the string "abcde" into "cstring" beginning at position two. After the replacement, it deleted the remaining characters in "cstring."

Applications

Lowercase All Characters in a String Except the First One

If you want to uppercase only the first letter of a string, you can do it with the STUFF() function as demonstrated by the following expression,

```
STUFF(UPPER( <string> ), 2, LEN( <string> ) - 1,;
LOWER(SUBSTR( <string> , 2)))
```

This a simple expression that is comparable to an equally simple concatenation expression, but it is useful to show how STUFF() can be used for concatenation operations. It is functionally equivalent to,

```
UPPER(LEFT( <string> , 1) + LOWER(SUBSTR( <string> ,2))
```

You can use it to your benefit if you need to uppercase the first letter of a substring within a larger string.

Replace All Occurrences of a Character with Another

You may on occasion have the need to replace all occurrences of one character with another. This can be done with a small program that uses STUFF() at its core to do the replacement of each character. The program below demonstrates how you can set this up and perhaps put it into practice,

```
* ——Program to replace all occurrences of char1 with char2.
char1 = "-"
char2 = "="
ctring = "==============="
DO WHILE char1 $ cstring
    cstring = STUFF(cstring, AT(char1, cstring), 1, char2)
 ENDDO
```

The algorithm works by first determining the next position in the target string to replace into. AT() does this. STUFF() then replaces the old character with the new one at the position of the old. The DO WHILE loop is used to continue the search and replace operation until there are no more occurrences of the target character to replace.

Replace Leading Blanks with Zeros

Suppose you have a character string with leading blanks in the form of a number, something that might be the result of using STR() on a numeric. For formatting purposes, you need to have leading blanks replaced by zeros. The following expression uses STUFF() to accomplish this.

```
STUFF( < string > ,1,LEN( < string > )-LEN(LTRIM( < string > )),;
REPLICATE("0",LEN( < string > )-LEN(LTRIM( < string > ))))
```

The key to this algorithm is determining the number of leading blanks. Subtract the total length of the string from the length of the LTRIM() of it. This number tells STUFF() how many characters to replace and REPLICATE() how many zeros to pass to STUFF() as the replacement string.

Right-Justify a Fixed Length Character String

On occasion you may want to right-justify the left-justified portion of a fixed length string essentially moving all of the trailing blanks to the beginning of the string. STUFF() can perform this in combination with SPACE(), LEN(), and TRIM() demonstrated by the following expression,

```
TRIM(STUFF( < string > , 1, 0, SPACE(LEN( < string > ) -;
LEN(TRIM( < string > )))))
```

This algorithm is a simple insertion and works first by determining the number of trailing blanks with the expression,

```
LEN( < string > ) - LEN(TRIM( < string > ))
```

This number is passed to SPACE() which in turn passes that number of blanks to STUFF(). STUFF() then inserts the blanks into the target string at the beginning and TRIM() deletes the trailing blanks.

Conclusion

In this short discussion, we have given you an overview of the STUFF() function, how it works, it general capabilities, and a few examples. We hope this discussion has illuminated you to the powers of this perplexing function. As you work with character string expressions, you may want to refer to the general capabilities discussion regularly until you master STUFF().

Framework II Meets the Mouse

by Kent Irwin

'Twas the night before Christmas,
and all through the house,
not a creature was stirring,
except for the mouse...'

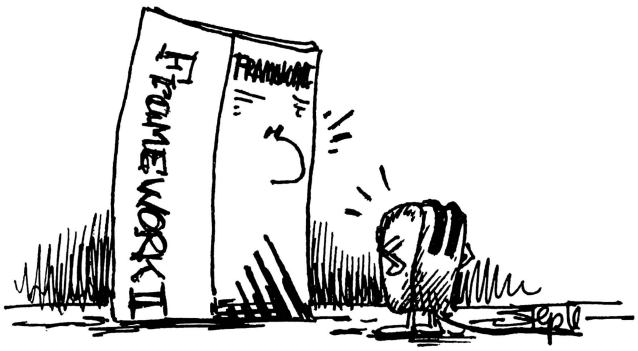
The mouse is a small electronic creature about the size of a dinner roll. It has one spherical "foot" upon which it glides, and two flattish "ears" that click when you press them. The long "tail" loops around your desk, and ultimately disappears into the back of your PC. Apart from being a rather cute conversation piece, the mouse can also be one of your most useful computer tools. Almost any program can be easier to use with a mouse. Hard-to-remember keys or commands can be selected by clicking a mouse button or by selecting from a pop-up menu. When used with Framework II, a mouse can help new PC users to learn the product more quickly and efficiently. To see how this is possible, let's bring the Microsoft Mouse and Framework II together.

Giving Your Mouse a Home

You won't find a mouse driver choice in the Setup program, since Framework II doesn't support a mouse. Using the mouse language that Microsoft provides with the mouse package, however, you can create your own mouse driver and custom pop-up menus. All you need to do is write a little program that tells the mouse what to do with Framework II. You will need the following files from the Microsoft Mouse disk (place them in your Framework II subdirectory): MOUSE.COM, MENU.COM, MAKEMENU.EXE. For complete details about the operation of each of those files, see the Microsoft Mouse manual.

To use the mouse, you won't need to write any FRED programs. You must, however, install a text mode screen driver (such as the IBM Color Text driver) using the Framework II Setup program. The mouse does not work well with a graphics screen driver. After you have made sure that you have an appropriate screen driver selected in Setup, follow these steps to create the mouse driver program,

1. Type the program in Figure 1 into a word frame labelled Fw.
2. **Export** Fw to an ASCII Text file.
3. **Quit** Framework II.
4. At the DOS prompt, type RENAME FW.TXT FW.DEF.
5. Run MAKEMENU and convert FW.DEF to FW.MNU.
6. Set up the batch file to run Framework II (Figure 2).
7. Run Framework II.



Putting the Mouse to Work

Now that you are in mouse-driven Framework II with the mouse driver installed, you can begin to investigate the mouse features. Initially, the left button performs an **Ins** and the right button performs an **Esc**. Press the left button and a menu pulls down. Moving the mouse moves the cursor just like the arrow keys do. Slide the mouse left and right to access other menus. Slide it up and down to choose a command. Press the left button again to execute the selected command. Press the right button to close the pull-down menus.

If you press both buttons together, a pop-up menu appears (see Figure 3). Select an option by moving the mouse up and down, and press either button to perform the option. The macros menu contains some handy macros that you can quickly execute just by selecting them from the menu. For example, the Abbreviations type out the indicated word or words and you need not type even the abbreviation itself.

The last two choices in the main menu are the most powerful. They allow you to redefine the function of each mouse button. Pressing both buttons always pops up the main mouse menu, but the operation of either button alone can be changed to one of the function keys, **Esc**, **Return**, **Uplevel**, **Dnlevel**, or **Scroll-Lock**. This way, you can adapt the mouse to whatever Framework II tasks you spend the most time on.

You can customize the mouse program (Figure 1) for your system. You can add or delete macros or abbreviations from the menus and as well, change the choices in the button definition menus. Use Framework II to edit the FW.DEF file, using the Microsoft Mouse Manual as a guide, and then follow steps 2 through 7 above. Note that the Extended Character codes (sometimes called Secondary Scan Codes) can be found in an appendix of the IBM BASIC manual.

Happy holidays, and keep those cookies away from the mouse!

Mouse Driver Program Listing

Figure 1.

```
; Framework II MS-mouse interface
;
;   by Kent Irwin
;      December 1, 1986
;
BEGIN lb,rb,bb,left,right,up,down,40,40
lb:    TYPE 0,82 ; Instruct (pull down a menu)
rb:    TYPE esc      ; Esc
bb:    EXECUTE menu1 ; open mouse menu
left:  TYPE 0,75
right: TYPE 0,77
up:    TYPE 0,72
down:  TYPE 0,80

menu1: MENU "BASIC Commands",5,20
       OPTION "Cancel Menu"
       OPTION "Macros",F1
       OPTION "Abbreviations",F2
       OPTION "Change Left Button",F3
       OPTION "Change Right Button",F4
       MEND

F1:    MENU "Macros",5,20
       OPTION "Cancel Menu"
       OPTION "Cut",C1              ; Alt-F7
       OPTION "Paste",C2            ; Alt-F8
       OPTION "Write text file",C3  ; Alt-W
       OPTION "Uppercase",C4        ; Alt-U
       OPTION "Lowercase",C5        ; Alt-L
       OPTION "Show Library",C6     ; Alt-F1
       OPTION "Memory Avail.",C7    ; Alt-F5
       OPTION "Type Date",C8        ; Alt-F3
       MEND

C1:    TYPE 0,110
C2:    TYPE 0,111
C3:    TYPE 0,17
C4:    TYPE 0,22
C5:    TYPE 0,38
C6:    TYPE 0,104
C7:    TYPE 0,108
C8:    TYPE 0,106

F2:    MENU "Abbreviations",5,20
       OPTION "Cancel Menu"
       OPTION "@eraseprompt",M1     ; "ep" and Alt-Backspace
       OPTION "@performkeys()",M2   ; "pf" and Alt-Backspace
       OPTION "@setdirectory",M3    ; "sd" and Alt-Backspace
       OPTION "@setselection",M4    ; "ss" and Alt-Backspace
       MEND

M1:    TYPE "ep",0,140
M2:    TYPE "pf",0,140
M3:    TYPE "sd",0,140
M4:    TYPE "ss",0,140

F3:      MENU "Change Left Button",3,20
```

```
            OPTION  "Cancel Menu"
            OPTION  "Ins",B1
            OPTION  "EDIT FORMULA F2",B2
            OPTION  "DRAG F3",B3
            OPTION  "SIZE F4",B4
            OPTION  "RECALC F5",B5
            OPTION  "EXTEND SELECT F6",B6
            OPTION  "MOVE F7",B7
            OPTION  "COPY F8",B8
            OPTION  "ZOOM F9",B9
            OPTION  "VIEW F10",B10
            OPTION  "Return",B11
            OPTION  "Lt=Uplv/Rt=Dnlv",B13
            MEND

B1:         ASSIGN one
B2:         ASSIGN two
B3:         ASSIGN three
B4:         ASSIGN four
B5:         ASSIGN five
B6:         ASSIGN six
B7:         ASSIGN seven
B8:         ASSIGN eight
B9:         ASSIGN nine
B10:        ASSIGN ten
B11:        ASSIGN eleven
B12:        ASSIGN twelve
B13:        ASSIGN uplv,dnlv

one:        TYPE 0,82
two:        TYPE 0,60
three:      TYPE 0,61
four:       TYPE 0,62
five:       TYPE 0,63
six:        TYPE 0,64
seven:      TYPE 0,65
eight:      TYPE 0,66
nine:       TYPE 0,67
ten:        TYPE 0,68
eleven:     TYPE enter
twelve:     TYPE esc
thirt:      TYPE 0,76
dnlv:       TYPE 0,70
uplv:       TYPE 0,69

F4:         MENU "Change Right Button",3,20
            OPTION "Cancel Menu"
            OPTION "Esc",D1
            OPTION "Scroll-Lock",D2
            OPTION "Return",D3
            MEND

D1:         ASSIGN ,twelve
D2:         ASSIGN ,thirt
D3:         ASSIGN ,eleven
; End of procedure.
```

Batch File Listing

```
REM INSTALL MOUSE DRIVER FOR FRAMEWORK II
MOUSE
MENU FW
FW
MENU OFF
```

Figure 2.

Change Summary for dBASE III PLUS Version 1.0—Part II

by Gail Kulikowsky

This is the second part of a multiple-part series that lists the anomalies reported for dBASE III PLUS version 1.0 and fixed in dBASE III PLUS version 1.1. They are listed here so that dBASE III PLUS version 1.0 users will have a clear idea of which anomalies have been corrected and which are still outstanding.

@ GET / RANGE

In dBASE III PLUS version 1.0, if you pressed **Space** in response to a RANGE error message and STATUS was ON, the message did not clear.

Now, pressing **Spacebar** clears the RANGE error message when STATUS is ON.

Not reported in ***TechNotes***.

BROWSE

In dBASE III PLUS version 1.0, BROWSE NOMENU locked out the cursor-control menu invoked by **F1** instead of the menu bar invoked with **F10** as documented.

Now, the NOMENU option correctly suppresses access to the BROWSE menu bar.

Not reported in ***TechNotes***.

CALL

In dBASE III PLUS version 1.0, typing WITH incorrectly when CALLing a routine froze the computer, requiring a warm boot.

Now, if you misspell WITH, dBASE III PLUS version 1.1 returns the error message "Syntax error."

Not reported in ***TechNotes***.

CATALOG (ASSIST)

In dBASE III PLUS version 1.0, attempting to select an index file in another subdirectory when a CATALOG was in use returned an inappropriate error message such as "File does not exist" or "Index file does not match database." This happened because the path stored in the Path field of the Catalog was not used to point to the location of the index file.

Now, when you select an index file from the **Set up:Database** file menu, the index file loads correctly if the path is specified in the Catalog.
Not reported in ***TechNotes***.

CONFIG.DB / DELIMITERS

In dBASE III PLUS version 1.0, you could SET DELIMITERS ON and OFF in Config.DB, but you could not set the delimiter character.

Now in dBASE III PLUS version 1.1, you can set a delimiter character in Config.DB.

Not reported in ***TechNotes***.

CONFIG.DB / PROMPT

In dBASE III PLUS version 1.0, setting PROMPT = <cstring> in Config.DB to a string of 20 characters or greater corrupted memory. If the string was longer than 20 characters, the error message "↑— truncated" appeared and the PROMPT was truncated to 20 characters.

Now you can specify a PROMPT of greater than 20 characters without affecting the performance of dBASE III PLUS. The length of the prompt is still 20 characters and you will still get the error message "↑— truncated" if you specify a longer prompt.

Not reported in ***TechNotes***.

CREATE SCREEN

In dBASE III PLUS version 1.0, if you used **Return** to insert blank lines when moving data down on the blackboard, the cursor could disappear below the status line and the only way to get it back was to press **Uparrow.**

Now in dBASE III PLUS version 1.1, when the cursor reaches the last line of the blackboard, it stays there while the data continues to move to higher-numbered lines.

Not reported in ***TechNotes***.

HELP

In dBASE III PLUS version 1.0, if HELP.DBS could not be located, the error message "Help text not found" flashed on the screen and disappeared too quickly to be read.

Now, the error message appears and you are prompted with the message to "Press any key to continue."

Not reported in ***TechNotes***.

INSERT

In dBASE III PLUS version 1.0, INSERT with SET CARRY ON did not add the proper record if you pressed **PgDn** instead of **Ctrl-W**. Pressing **PgDn** appended information from the record following the INSERTed record instead of information from the record preceding it.

Now if you INSERT, the new record will have the information from the current record and not the current record plus one when you SET CARRY ON and press **PgDn** to terminate.

For example, if you INSERT at record 6, the new record 7 will have the information from record 6 and not the old record 7 as before.

Not reported in ***TechNotes***.

LIST/DISPLAY TO PRINT

In dBASE III PLUS version 1.0, LIST and DISPLAY TO PRINT did not print the last record sent to the printer until the printer was taken off-line or received another printable line.

Now in dBASE III PLUS version 1.1, all records sent to the printer with LIST or DISPLAY TO PRINT are printed. This fix, however, introduced another problem. DISPLAY and LIST NEXT 1 directed to the printer with the TO PRINT clause now send an extra carriage return line feed pair. This is different from both dBASE III PLUS version 1.0 and dBASE III version 1.1. In dBASE III version 1.1, for example, DISPLAY TO PRINT sent a carriage return line feed both before and after the DISPLAY output. A DISPLAY not directed to the printer with the TO PRINT clause did not send a trailing carriage return line feed to the screen.

Not reported in ***TechNotes***.

MIN()

In dBASE III PLUS version 1.0, the MIN( ) function returned a numeric overflow if either of the arguments were negative numbers and different lengths. For example,

```
? MIN(-1, -2)
```

returned -2 as you would expect. However,

```
? MIN(1, -2)
```

returned a numeric overflow. Curiously,

```
? MIN(01, -2)
```

returned the expected result of -2 since both arguments had the same length.

Now, MIN( ) returns an appropriate answer no matter what the difference in the lengths of the arguments.

Not reported in ***TechNotes***.

SAVE

In dBASE III PLUS version 1.0, using this syntax:

```
SAVE TO <memfile>, mvar1, mvar2, mvar3
```

resulted in the correct error message, but the memory file was left open and you could not close it without QUITing dBASE III PLUS.

Now, when the SAVE TO fails, the memory file is closed.

Not reported in ***TechNotes***.

SET CATALOG TO

In dBASE III PLUS version 1.0, SETting CATALOG TO a filename that was twelve or more characters long caused the computer to freeze or returned you to the DOS prompt.

Now, if you specify a Catalog filename in excess of eight characters, dBASE III PLUS truncates the filename to the first eight characters.

Not reported in ***TechNotes***.

SET DEVICE TO PRINT / @ row,col

In dBASE III PLUS version 1.0, if you executed a @ <row>, <column> command after you SET DEVICE TO PRINT, the screen line would still CLEAR.

Now if you execute an @ <row>, <column> with no argument and the DEVICE is SET TO PRINT, the output of the @ <row>, <column> is sent to the printer and does not affect the screen. Be aware that if the row and column positions of this statement are less than those of the last @...SAY directed to the printer, you will get a page eject.

Not reported in ***TechNotes***.

SET FORMAT TO

In dBASE III PLUS version 1.0, if you opened a format file before you USEd the accompanying database file, you could not CLOSE the FORMAT file without QUITing dBASE III PLUS. In addition, when you used a full-screen command such as APPEND or EDIT, the format file did not come up. You could still SET FORMAT TO the same format file and use it without error. If, however, you attempted to MODIFY COMMAND with the format file, the error message "File is already open" was returned.

Now, you can SET FORMAT TO a format file before USEing the database file and close the format file without QUITting dBASE III PLUS.

Not reported in ***TechNotes***.

SIGMA 400 Color Board

In dBASE III PLUS version 1.0, the SIGMA 400 color graphics board caused your computer to freeze unless you ran NMI.COM using with this syntax:

```
NMI OFF
```

before loading dBASE III PLUS.

Now if you have a SIGMA 400 color graphics board in your computer, you can load dBASE III PLUS version 1.1 without running NMI.COM first.

Not reported in ***TechNotes***.

USE ?

In dBASE III PLUS version 1.0, if you selected a database file from a USE ? invoked by a program and ESCAPE was SET OFF, pressing **Esc** aborted the program with the error message "Called from - < program name > Cancel Ignore, Suspend? (C,I,S)."

Now in dBASE III PLUS version 1.1, you can SET ESCAPE OFF, call a Catalog query from within a command file, and press **Esc** to terminate the Catalog query without receiving a command file termination error message.

Not reported in ***TechNotes***.

ZAP

In dBASE III PLUS version 1.0, if you ZAPped a database file with more than one open index file, your next attempt to USE the database file with the index file list produced the error message "Index damaged. REINDEX should be done before using data."

Now, you can ZAP a database file with multiple index files in USE and then USE them without receiving an error message.

Not reported in ***TechNotes***.

Sending Control Codes to the Printer

by Kenneth N. Getz, Larry Abare, and Karen Robinson

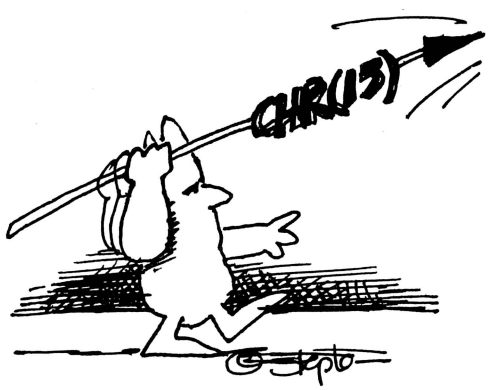
One of the frequently asked questions of the Software Support Center concerns how to send control codes to the printer from within dBASE III PLUS. We usually advise callers to use the ? or @...SAY command to send the codes to the printer. If you change printing fonts often (condensed to extra wide to back to condensed), repeatedly typing in long strings of control codes can be a tedious and sometimes confusing process. The two programs presented here, SetPrint.ASM and PatchPrt.PRG, allow you to store up to ten sets of printer control codes in a data table and send these codes to your printer from dBASE III, dBASE III PLUS, or DOS.

The main routine, SetPrint.ASM, is an assembly language program that stores the codes in a data table and sends them to your printer. You can RUN it as a .COM file or LOAD and CALL it as a .BIN file. The supporting program, PatchPrt.PRG, is a dBASE III PLUS program that provides a way for you to add, edit, and change codes stored in the data table without having to reassemble and link SetPrint.ASM.

Setprint.ASM

SetPrint stores up to ten different printer control strings in a data table. Each string can have as many as 26 codes. You can enter these codes directly into the table or through the dBASE III PLUS program, PatchPrt.PRG.

To use SetPrint from within dBASE III or dBASE III PLUS, your computer must have at least 320K of RAM. To assemble SetPrint.ASM as a .COM or .BIN file, you need the IBM or Microsoft Macro Assembler (MASM.EXE), version 2.0 or higher. The smaller assembler, ASM.EXE, will not work since it does not support macros or conditional assembly, both of which are used in SetPrint.ASM. You will also need the programs LINK.EXE and EXE2BIN.EXE, included on the DOS Supplemental Programs disk with version 2.0 and higher.



SetPrint accepts one parameter, either a ? or a number between 0 and 9 inclusive. If you are executing it as a .BIN file, the parameter is a character string. If you are executing it as a .COM file from DOS or dBASE, the parameter is a literal. For example,

```
LOAD SetPrint
CALL SetPrint with "3"
```

or

```
RUN SetPrint 3
```

or

```
A > SetPrint 3
```

When you enter a number (0 through 9), SetPrint sends the control codes corresponding to that table position to the printer. The example above sends the fourth set of control codes. There are no messages displayed to indicate whether or not the printer accepted the codes. So, if your printer is off when you execute SetPrint, the codes are simply ignored. Furthermore, when the codes are sent to the printer, you will see the changed print font only when you print a document.

You can use the ? symbol as a parameter to display a list of the codes stored in the table. If you are displaying this list from within dBASE III PLUS, SET STATUS OFF first in order to prevent the status bar from moving up the screen. In addition, to make sure that all of the control codes display, do not use ANSI.SYS as a device driver. If you use ANSI.SYS, the list may be incomplete since non-printable characters, such as ASCII 27, do not display.

In addition to updating the table through PatchPrt, you can also change the data table directly. The table of codes is stored in SetPrint and by default is null-filled to 26 characters. In the program code, the table looks like this,

```
table   db      26 dup(0)
        db      26 dup(0)
        db      26 dup(0)
        db      26 dup(0)
        db      26 dup(0)
        db      26 dup(0)
        db      26 dup(0)
        db      26 dup(0)
        db      26 dup(0)
        db      26 dup(0)
```

The table contains ten rows, one row per string of control codes. When you enter the printer control codes, use ASCII decimal value for each character separating them by commas. If your code sequence is less than 26 decimal codes long, pad the string with nulls (00) out to 26 places. Option 0 in the table below is the Epson condensed print codes, Esc 15. The command 24 dup(0) tells the program to attach 24 null characters to the end of the sequence. Be sure to use this command to pad the printer control codes strings that you enter.

```
db      27,15,24  dup(0)
db      26  dup(0)
db      26  dup(0)
db      26  dup(0)
db      26  dup(0)
db      26  dup(0)
db      26  dup(0)
db      26  dup(0)
db      26  dup(0)
db      26  dup(0)
```

Each time you change the table of printer codes, SetPrint must be reassembled and linked.

PatchPrt.PRG

PatchPrt is a dBASE III PLUS program that provides an easy method for entering and updating the table of printer control codes without having to edit, reassemble, and link SetPrint.ASM. (PatchPrt runs only in dBASE III PLUS because it uses commands and functions that are not available in dBASE III.) When you use this program, make sure that the files PatchPrt.PRG and SetPrint (.BIN or .COM) are in the same subdirectory. The DOS file DEBUG.COM must also be available. Either copy it into the same subdirectory as the other files or set a DOS path to its location.

Enter the printer control codes as literal characters or as ASCII decimal values. ASCII decimal values must be entered as a three-digit numbers delimited with braces, such as {027} for **Esc**. Enter each code from left to right without separating them with spaces or commas. For non-printable characters, such as **Esc**, always use the ASCII decimal value. For example, on an Epson printer, the code sequence for double-strike is **Esc** G. You can enter this in either of the following ways:

{027}G

or

{027}{071}

Each sequence is limited to 26 codes. If you enter more, PatchPrt truncates them. This limitation refers to the actual number of ASCII codes sent, not to the number of characters used to send the control codes. The letter "A" is considered a single code, for example; its decimal equivalent, {065}, is also considered a single code.

When you execute PatchPrt, it displays a description of the printer control codes previously entered through PatchPrt. If you enter the control codes directly into the table, a description does not display. PatchPrt prompts you to select a sequence to edit or to type Q (quit). Once the printer control codes and descriptions are entered, PatchPrt translates the control codes to their hexadecimal values and writes these values to a text file using SET ALTERNATE. The text file is then used by DEBUG.COM to update the data table in SetPrint. The program finishes by reloading the newly updated module using the LOAD command.

Setup

To set up the programs, enter the code for both PatchPrt.PRG and SetPrint.ASM using MODIFY COMMAND or any text editor that generates standard ASCII text files. If you plan to execute SetPrint from DOS or RUN it from dBASE III, assemble it as a .COM file. If you are going to use the LOAD/CALL command in dBASE III PLUS, assemble it as a .BIN file.

To use SetPrint as a .COM file, change the second line of the program code to read as follows:

```
COM EQU 1
```

To use it as a .BIN file, the second command line should read:

```
COM EQU 0
```

To assemble SetPrint.ASM as either a .COM or .BIN file, the following files must be available, either in the same subdirectory as SetPrint.ASM or through the DOS PATH command: MASM.EXE, LINK.EXE, and EXE2BIN.EXE. Follow the steps below to assemble SetPrint.ASM as either a .COM or .BIN file.

1. From the DOS prompt, type:

   ```
   MASM SetPrint.ASM;
   ```

   This creates an object file (SetPrint.OBJ).

2. Then, type:

   ```
   LINK SetPrint;
   ```

   This creates SetPrint.EXE.

3. To create a .COM file, type:

   ```
   EXE2BIN SetPrint.EXE SetPrint.COM
   ```

   To create a .BIN file, type:

   ```
   EXE2BIN SetPrint.EXE
   ```

4. Last, delete SetPrint.EXE:

   ```
   ERASE SetPrint.EXE
   ```

The programs presented here will save you time and increase your efficiency by allowing you to store and call your ten most-often-used printer control codes from any of three places: dBASE III, dBASE III PLUS, and DOS. If your office has several different kinds of printers, you can create a custom version of SetPrint for each printer type.

The usefulness of SetPrint is not limited to dBASE applications. Because you can also run it from the DOS level, it's actually a generic utility that can be used alone or with other software programs.

Code

```
; Program ....: Setprint.ASM
; Author .....: Ken Getz
; Date .......: December 1, 1986
; Version ....: dBASE III, Developer's Release, dBASE III PLUS
; Note(s) ....: Sends preplanned printer control strings to the
;               printer either from DOS or from within dBASE.
;

CODE        SEGMENT     BYTE PUBLIC         'CODE'

COM         EQU         0               ; Should read 'COM EQU 1'for COM file.
MAX         EQU         26
TOTAL       EQU         10
CR          EQU         13
LF          EQU         10

write1      MACRO       char            ; Macro to write a single character.
            PUSH        AX
            PUSH        DX
            MOV         DL,char
            MOV         AH,2
            INT         21h
            POP         DX
            POP         AX
            ENDM

SETPRINT    PROC        FAR
            ASSUME      CS:code,DS:code

IF          COM                         ; Indicates where to look
            ORG         05Dh            ; for param if COM file.
param       LABEL       byte
            ORG         100h
ENDIF

START:      JMP         ENTRY

table       db          26  dup(0)       ; Empty table for printer codes.
            db          26  dup(0)
            db          26  dup(0)
            db          26  dup(0)
            db          26  dup(0)
            db          26  dup(0)
            db          26  dup(0)
            db          26  dup(0)
            db          26  dup(0)
            db          26  dup(0)

ENTRY:
            PUSH        CX              ; Save working environment.
            PUSH        SI
            PUSH        DX
            PUSH        BX
            PUSH        AX
            PUSH        DS

IF          COM                         ; If a .COM file,
            MOV         AL,param        ; get the parameter.
            CMP         AL,32           ; Check for a parameter.
```

```
            JG          PARAM_OK1       ; If it's there, continue.
            JMP         EXIT            ; If not, quit.

ELSE
            MOV         AX,[BX]         ; If a .BIN file, get the
ENDIF                                   ; parameter from dBASE.

PARAM_OK1:
            PUSH        CS
            POP         DS              ; Force DS to be the same as CS.
            CMP         AL,'?'          ; If the parameter is '?',
            JNE         NOT_HELP        ; display all options, otherwise,
            JMP         DISPLAY_ALL     ; check for the specific parameter.
NOT_HELP:
            CMP         AL,'0'          ; Is parameter greater than '0'?
            JGE         RANGE_CHECK
            JMP         EXIT            ; If not, quit.
RANGE_CHECK:
            CMP         AL,'9'          ; Is it less than '9'?
            JLE         PARAM_OK3
            JMP         EXIT            ; If not, quit.
PARAM_OK3:                              ; Convert parameter to an integer value
            SUB         AL,48           ; by subtracting the ASCII value of '0'.
            XOR         AH,AH           ; Zero out top byte of AX.
PARAM_OK2:
            MOV         CX,MAX          ; Store length of printer code to CX.
            MUL         CX              ; Get offset into table by
                                        ; multiplying the offset (AX)
                                        ; by the length of the code (CX).
            MOV         SI,OFFSET table ; Get the offset of the table of printer codes.
            ADD         SI,AX           ; Add that to the original offset.
LOOPER:                                 ; Send the codes to the printer.
            MOV         DL,[SI]         ; Store a single character in DL.
            MOV         AH,5            ; Call service 5 of
            INT         21h             ; INT 21h.
            INC         SI              ; Go to next character
LOOP LOOPER                             ; until CX = 0.
            JMP         EXIT
DISPLAY_ALL:                            ; Loops to display all codes.
            MOV         DI,OFFSET table
                                        ; Store the location of the
                                        ; table in DI.
            MOV         CX,TOTAL        ; Store the total number of
                                        ; strings in CX.
            write1      CR              ; Call the macro 'write1'to
                                        ; write a single character.
            write1      LF
OUTER_LOOP:
            XOR         BX,BX           ; Zero out the BX register.
            MOV         DX,TOTAL        ; Store the total number of
                                        ; strings in DX.
            SUB         DX,CX           ; Compute the string number (0 - 9).
            ADD         DX,48           ; Convert to character
                                        ; by adding ASCII '0'.
            write1      DL              ; Write the string number.
            write1      ':'
            write1      ' '
INNER_LOOP:
            CMP         BYTE PTR [DI],0
                                         ; Search for ASCII NULL to see
                                         ; if at end of string.
            JE          INC_DI           ; If so, go on to the next string.
            MOV         DL,BYTE PTR [DI]
```

```
                                        ; Otherwise, move the code into DL
            write1      DL              ; and write it out.
INC_DI:
            INC         BX              ; Go to next string.
            INC         DI
            CMP         BX,MAX          ; See if we're at the last one.
            JL          INNER_LOOP      ; If not, go back for more.
            write1      CR
            write1      LF
            LOOP        OUTER_LOOP
EXIT:
            POP DS                      ; Restore the environment.
            POP         AX
            POP         BX
            POP         DX
            POP         SI
            POP         CX
IF          COM                         ; If this is a COM program, quit to DOS.
            MOV         AX,4C00h
            INT         21h
ELSE
            RET                         ; Otherwise, return to dBASE.
ENDIF
SETPRINT    ENDP
CODE        ENDS
            END         START
```

```
* Program ....: Patchprt.PRG
* Author .....: Larry Abare
* Date .......: December 1, 1986
* Version ....: dBASE III Plus, Developer's Release
* Note(s) ....: Utility to modify printer codes in assembled .BIN file. Asks user for string
*               to modify, then characters to replace. Does NO error checking on validity
*               of printer codes. Assumes that DEBUG.COM is available (in same directory
*               or PATH).
*
SET STATUS OFF
SET SAFETY OFF
SET TALK OFF
SET BELL OFF
SET ALTERNATE TO Prtcodes.txt
* ---Modifiable parameters
infile = "Setprint.BIN"
choice = SPACE(1)
* ---Length of all the printer codes
codelen = 26
STORE SPACE(35) TO s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,sc0,sc1,sc2,sc3,sc4,sc5,sc6,sc7,sc8,sc9
IF FILE("Codes.MEM")
   RESTORE FROM Codes ADDITIVE
ENDIF
* ---Offset of printer table in .BIN file
table = 3
full_pad = REPLICATE("00 ",codelen)
CLEAR
? "Each sequence may be up to " + LTRIM(STR(codelen, 2, 0))
?? " characters long if needed."
? "You must use the ASCII representation or decimal equivalent of the byte "
? "to be sent to the printer."
? "Non-displayable characters such as CHR(15) or Escape must be encased in "
? "soft brackets, and consist of 3 digits. "
? "EXAMPLE: An asterisk may be entered as * or {042}, Escape as {027}, etc.."
```

```
@ 7,0 to 7,79
@ 13,0 to 13,79
@ 19,0 to 19,79
DO WHILE .T.
   @ 8, 1 CLEAR TO 12,79
   @ 8, 1 SAY "0: "+S0+" 5: "+S5
   @ 9, 1 SAY "1: "+S1+" 6: "+S6
   @ 10,1 SAY "2: "+S2+" 7: "+S7
   @ 11,1 SAY "3: "+S3+" 8: "+S8
   @ 12,1 SAY "4: "+S4+" 9: "+S9
   @ 14,0 CLEAR TO 18,79
   @ 20,0 CLEAR TO 24,79
   code_num = SPACE(1)
   @ 20,0 SAY "Enter the number (0-9) to assign to this sequence."
   @ 21,0 SAY "Enter Q to quit... "
   DO WHILE .NOT. code_num $ "0123456789Q"
      code_num = SPACE(1)
      @ 21,20 GET code_num;
              PICTURE "!"
      READ
   ENDDO
   IF code_num = "Q"
      EXIT
   ENDIF
   STORE s&code_num TO change
   @ 14,20  SAY "String Description: " + s&code_num
   @ 16,1   SAY sc&code_num
   @ 21,0   SAY "Enter String Description"
   @ 22,0   GET s&code_num
   READ
   @ 23,0 SAY "Do you want to change this control codes sequence [Y/N]";
          GET choice;
          PICTURE "Y"
   READ
   IF UPPER(choice) = "N"
      LOOP
   ENDIF
   @ 21,0 CLEAR
   @ 21,0 SAY "Enter Printer Code Sequence " + REPLICATE(SPACE(1),30)
   ACCEPT "-> " TO p_string
   sc&code_num  = p_string
   pstring_len  = LEN(p_string)
   counter      = 1
   xtra_count   = 0
   hex          = "0123456789ABCDEF"
   max_count    = 0
   endstring    = ""
   DO WHILE counter <= pstring_len
      nextchar = STR(ASC(SUBSTR(p_string, counter, 1)), 3, 0)
      IF nextchar = "123"
         IF STR(ASC(SUBSTR(p_string, counter + 4, 1)), 3, 0) = "125"
            nextchar   = SUBSTR(p_string, counter + 1, 3)
            counter    = counter + 4
            xtra_count = xtra_count + 4
         ENDIF
      ENDIF
      temp_c = SUBSTR(hex, INT(VAL(nextchar) / 16) + 1, 1) +;
               SUBSTR(hex, MOD(VAL(nextchar), 16) + 1, 1)
      endstring = endstring + " " + temp_c
      counter   = counter + 1
      max_count = max_count + 1
      IF max_count > 26
         ? CHR(7)
```

```
        @ 20,0 CLEAR TO 23,79
        @ 21,8 SAY "The string has exceeded the 26 byte " + "limit... Please Re-Enter"
        ?
        WAIT
        LOOP
      ENDIF
      @ 22 + IIF(counter > 77, 1, 0), (counter - IIF(counter > 77, 80, 0)) + 2 SAY "."
    ENDDO
    pad = " " + SUBSTR(full_pad,1,3 * (codelen + xtra_count -
            pstring_len))

 mlocation1 = SUBSTR(hex, INT(VAL(m_location) / 16) + 1, 1) +;
              SUBSTR(hex, MOD(VAL(m_location), 16) + 1, 1)
   m_location = STR(VAL(code_num) * codelen + table + 15, 3, 0)
   mlocation2 = SUBSTR(hex, INT(VAL(m_location) / 16) + 1, 1) +;
              SUBSTR(hex, MOD(VAL(m_location), 16) + 1, 1)
   newstring1 = SUBSTR(endstring + pad, 1, 45)
   newstring2 = SUBSTR(endstring + pad, 46, 45)
   SET CONSOLE OFF
   SET ALTERNATE ON
   ? "E1" + mlocation1 + newstring1
   ? "E1" + mlocation2 + newstring2
   SET ALTERNATE OFF
   SET CONSOLE ON
ENDDO
SAVE ALL LIKE S* TO Codes
SET CONSOLE OFF
SET ALTERNATE ON
? "W"
? "Q"
?
CLOSE ALTERNATE
SET CONSOLE ON
CLEAR
verify = "N"
? "NOTE: This program assumes that Escape Sequences were entered correctly!"
?
? " Proceeding will write only new sequences to "+UPPER("&infile")+" Module"
? " and leave non-selected sequences intact."
?
@ 8,0 SAY "Enter [Y/N] to continue... "
@ 8,32 GET verify;
PICTURE "Y"
READ
IF UPPER(verify)="Y"
   CLEAR
   @ 1,1 SAY "Updating " + UPPER("&infile")
   SET CONSOLE OFF
   RUN DEBUG &infile < Prtcodes.TXT
   SET CONSOLE ON
   LOAD &infile
   DELETE FILE Prtcodes.TXT
ENDIF
CLEAR
SET STATUS ON
SET SAFETY ON
SET TALK ON
SET BELL ON
* EOP Patchprt.PRG
```

File Handle Status

by Kenneth N. Getz and Brenda Johnson-Grau

If you have encountered the error message "Too many files are open" while developing an application program, you probably have endured some frustration debugging the problem. This is particularly the case where you have a large application program comprised of many file access modules. Your problem could be related to the DOS configuration set in CONFIG.SYS or the result of program control and file management.

The following program, Handles.ASM, is an assembly language program that you LOAD and CALL allowing you to check the number of file handles available at any given time from within dBASE III PLUS. It works by subtracting the number of file handles used from 15, the maximum limit imposed by DOS after dBASE III PLUS has been loaded. So, it assumes your FILES setting in CONFIG.SYS is 20. If your "Too many files are open" error condition is caused by a problem with CONFIG.SYS, this utility is not useful. Additionally, if have LOADed and CALLed ADDFILES.BIN from the *dBASE Programmer's Utilities* to extend the total number of file handles available under DOS 3.0 AND above to 20, you need to add five to the result of Handles.

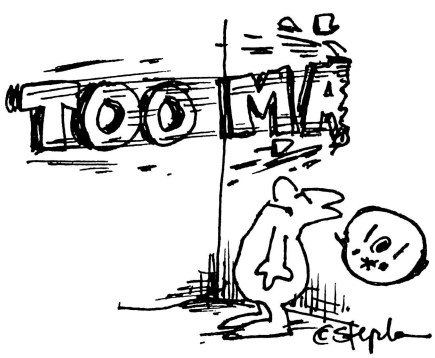
Creating and Using the Handles Program

1. Type the program, Handles.ASM, listed below using an ASCII text editor (the dBASE Word Processor, for example).
2. Using the IBM or Microsoft Macro Assembler, type these command lines from the DOS prompt:

```
MASM HANDLES
LINK HANDLES
EXE2BIN HANDLES.EXE
```

This sequence of commands creates a .BIN file that you can LOAD and CALL from within dBASE III PLUS or the Developer's Release.

Although .ASM programs given in ***TechNotes*** are usually written for conditional assembly to run either from DOS (as .COM files) or from dBASE III PLUS (as .BIN files), this program is meaningless except as a direct LOAD/CALL. There are, therefore, no instructions given for conditional assembly.



For more information on using assembly language programs with dBASE III PLUS, see Ralph Davis' article "Interfacing Assembly Language Routines with dBASE" in the September 1985 issue of ***TechNotes***.

Be sure to SET STATUS OFF before LOADing or CALLing Handles.BIN, otherwise, the output will be unreadable. From within dBASE III: the Developer's Release or dBASE III PLUS (the two versions of dBASE that include the CALL and LOAD commands), type this command:

LOAD Handles

Whenever you want to see how many files are available, type:

CALL Handles

The screen display shows the number of files that are available. When debugging, you can LOAD Handles once and then CALL it each time you need to verify the number of open files.

For more information on CALL and LOAD, read the reference section of *Using dBASE III PLUS*.

```
; Program ...: Handles.ASM
; Author ....: Ken Getz and Leonard Zerman
; Date ......: December 1, 1986
;
; .BIN file to be loaded from within dBASE III PLUS or the Developer's Release,
; to print out the number of available file handles for debugging purposes.
; Uses service 44H of INT 21H to return status of first 20 file handles. The
; interrupt returns a 6 if the file was unavailable (currently not in use);
; so, all occurrences of unavailable files are counted and returned to the user.
; (Note: This simple program assumes the line FILES=20 is in CONFIG.SYS.)
;
CODE   SEGMENT   BYTE PUBLIC   'CODE'

CR   EQU   13
LF   EQU   10

write1     MACRO      char     ; Macro to write a single character.
      PUSH ..... AX
      PUSH ..... DX
      MOV  ..... DL,char
      MOV  ..... AH,2
      INT  ..... 21h
      POP  ..... DX
      POP  ..... AX
      END  ..... M

HANDLES PROC  FAR
     ASSUME   CS:code,DS:code

START:   JMP  ENTRY
MSG  DB  CR,LF,'Files currently available: $'
ENTRY:
     PUSH CX          ; Save environment.
     PUSH DI
     PUSH DX
     PUSH BX
     PUSH AX
     PUSH DS
     PUSH CS          ; Make CS and DS have the same value.
     POP DS
     XOR DI,DI        ; Zero out DI.
```

```
     MOV     CX,19
LOOPER:
     MOV     AH,44h            ; IOCTL service.
     XOR     AL,AL             ; Zero out AL.
     MOV     BX,CX             ; BX := file handle to check.
     INT     21h               ; Call DOS.
     CMP     AL,06             ; Check for "error" -- file not available.
     JNE     NO_ERROR          ; If no error, jump over increment.
     INC     DI                ; Add one to counter.
NO_ERROR:
     LOOP    LOOPER            ; Loop back to use all possible file handles.
     MOV     DX,OFFSET MSG     ; Get message to print out.
     MOV     AH,9              ; Print message service.
     INT     21h               ; Call DOS.
     MOV     AX,DI             ; AX := binary number to convert to decimal.
     CALL    DEC               ; Convert to decimal and print.
     WRITE1 CR                 ; Write carriage return.
     WRITE1 LF                 ; Write linefeed.
EXIT:
      POP    DS                ; Restore all registers.
     POP     AX
     POP     BX
     POP     DX
     POP     DI
     POP     CX
     RET
HANDLES    ENDP


; Recursive procedure to convert a binary number stored in AX to decimal
; and print it.
;
;    by Robert Doolittle
;
DEC   PROC   NEAR
      MOV    CX,-10
DEC3: PUSH   DI
      MOV    DI,-1
DEC1: INC    DI
      ADD    AX,CX
      JB     DEC1
      SUB    AX,CX
      XCHG   AX,DI
      OR     AX,AX
      JZ     DEC2
      CALL   DEC3
DEC2: ADD    DI,30h
      MOV    DX,DI
      MOV    AH,2
      INT    21h
      POP    DI
      RET
DEC   ENDP
CODE  ENDS
      END    START
```

User-Defined Date and Character Fill Functions

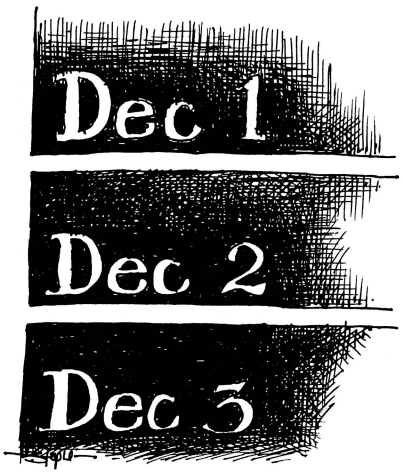
by Jack Hirsch

If you are working on a spreadsheet where you need to fill a region with a range of dates, you have probably found that the **@fill** function does not work with the date data type. For example, you may have an application that lists activities for every day of an entire year, one column with all the dates from January 1 to December 31. Naturally, you don't want to enter all 365 dates manually. To make this operation easier, you can use the following user-defined function, FillDate, to fill a specified region with dates in chronological order. In addition, you can modify FillDate into FillChar, to fill a region with a range of ASCII characters.

Filling a Region with Dates

To set up FillDate as a user-defined function accessible from anywhere on the desktop, create an empty word frame in the User-defined functions section of the Library and label it FillDate. Then, press **EDIT FORMULA (F2)**, **ZOOM (F9)**, and enter the following formula,

```
; FillDate.FW2
;
;    by Jack Hirsch
;       December 1, 1986
;
; Function to fill a specified region with dates in chronological order
;
; Syntax: @FillDate(month, day, year, inc, region)
;
; Declare and initialize a variable.
@local(day_counter),
day_counter := @item2,
; Loop until region is filled.
@while(@get(@item5) <> #NULL!,
    ; Using user supplied parameters, put in the dates.
    @put(@item5, @date(@item3, @item1, day_counter)),
    ; Using the parameter @item4, increment the counter.
    day_counter := day_counter + @item4,
    ; Skip to the next cell in the region.
@next(@item5)
)
```



To call FillDate, use the following general syntax,

@FillDate(month, day, year, inc, region)

Pass the parameters month, day, year, and inc as numerics. Pass the region parameter as a standard region reference. To demonstrate FillDate, assume you want to fill the region in a spreadsheet from cell A1 to A31 with chronologically ordered dates from December 1 through December 31, 1986. To do this, create a sample spreadsheet and label it Test. Next create an empty word frame. Label it Demodate, press **EDIT FORMULA (F2)** and enter the following calling formula:

@FillDate(12, 1, 1986, 1, Test.A1:Test.A31)

To see the function execute, highlight Demodate and press **RECALC (F5)**. The region will fill with the requested dates. Save Demodate and you can reuse it by changing the parameters. For example, you can produce a weekly or biweekly sequence by changing the value of the increment to 7 or 14. To list the dates for a pay day that occurs every two weeks, you could use this formula,

@FillDate(1, 1, 1987, 14, Pay Record.A1:Pay Record.A26)

Calling Filldate from within a Spreadsheet

If you prefer, you can place the formula that calls FillDate directly into a spreadsheet cell instead of into a separate word frame. Make sure, however, that you place the formula in a cell that is not included in the region that you are filling with dates.

The spreadsheet name is optional with this approach. Also, you may want to include a command to suppress any display from being returned to the calling cell. For example,

```
@FillDate(1, 1, 1987, 7, A1:BA1),
; This line returns one blank space in order to suppress the
; constant returned from FillDate.
@return(" ")
```

This formula generates the dates for every seven days, beginning with January 1, 1987, across the first row of the spreadsheet. As this demonstrates, FillDate works equally well with horizontal references.

Filling a Region with Characters

FillDate can be easily modified to help with another kind of fill function. This one involves filling a specified region with alphabetic characters.

To change FillDate, **COPY (F8)** it and label the new frame FillChar. Press **EDIT FORMULA (F2)** and **ZOOM (F9)** and change its formula to,

```
; FillChar.FW2
;
;   by Jack Hirsch
;      December 1, 1986
;
; Function to fill specified region with alpha characters
;
; Syntax:   @FillChar(ASCII code, inc, region)
;
; Declare and initialize a variable.
@local(char_counter),
char_counter := @item1,
; Loop until region is filled.
@while( @get( @item3 ) <> #NULL!,
    ; Using user supplied parameters, put in the characters.
    @put(@item3, @chr(char_counter)),
    ; Using the parameter @item4, increment the counter.
    char_counter := char_counter + @item2,
    ; Skip to the next cell in the region.
    @next(@item3),
)
```

Next, create an empty word frame on the desktop, label it Demochar, press **EDIT FORMULA (F2)**, and enter the following formula,

```
@FillChar(65, 1, Test.Cabinet)
          ↑
          └─ ASCII code for letter A
```

Now create a 26-record database called Test. Label the first column Cabinet. This column will hold the letters of the alphabet, A through Z. Highlight Demochar and press **RECALC (F5)**. The column Cabinet will fill with the letters from A through Z.

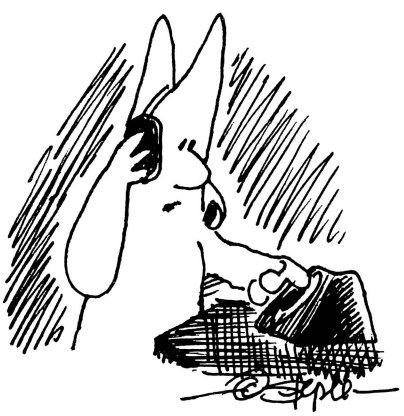
Pass the first parameter as the code for any ASCII character up to the 255, the maximum value in the IBM Extended Character Set.

RBBS Dialer

by Jack Hirsch

With many information services such as CompuServe, The Source, and electronic bulletin board (RBBS) systems available, you can accumulate a sizable list of regularly called numbers in a short period of time. If this is happening to you, developing a good information tracking system for bulletin boards quickly becomes a priority.

Framework II does provide a facility to satisfy your need for easy access to regular services on the main Telecommunications menu. This facility, however, has several limitations if you have a large number of services that you access. First, you may regularly call more than nine services and the Telecommunications menu provides for only nine quick access services. Second, you may want to add or change the list of services available while on the desktop without having to run the Setup program from DOS, as is the case with the services listed on the Telecommunications main menu. Lastly, you may want to track information about each RBBS you access and information about each access, such as date and time of last call. The program presented here, TCM Dialer, grew mainly from the need to circumvent these limitations. TCM Dialer can help you keep track of all your listings and provide you with the same kind of fast-and-easy access available through Framework II's Telecommunications menu. Additionally, you can modify or enhance it to meet your individual telecommunicating needs.



The RBBS Database Frame

TCM Dialer works on a Framework II database frame called RBBS that has the following fields,

Field	Description
Bulletin Board	RBBS/information service
City	City where RBBS is located
State	State where RBBS is located
Phone	Access phone number
Sysop	System operator's name
Board	RBBS Software
TCM File	.TCM filename
Last Access	Date of last logoff
Log Off	Time of logoff
Days Since	Number of days since last logoff

To setup RBBS, create an empty database frame with at least 12 columns. The number of rows is up to you. RBBS should have as many rows as you will have entries, though any number will do since you can add or delete rows after you have entered the data. Label each column with the field names listed above. You can **SIZE (F4)** the columns now or wait until you have entered the data.

There is only one field in the database frame that contains a formula, Days Since (the number of days since you last logged off the service). Position the cursor on this field and, press **EDIT FORMULA (F2)**, **ZOOM (F9)**, and enter the following formula,

```
@set(Days Since,
   @list(
      @if(@iserr(@sumdate(Last Access, 1)),
            "No access",
         @diffdate(@datetime(@date, @time), Last Access)
      )
   )
)
```

This formula calculates the number of days since your last logon to a particular bulletin board.

Adding and Deleting New Records

Except for the four fields listed below, you can add, edit, or delete data in the RBBS frame in the same way as in any Framework II database frame. The exceptions are:

1. Last Access and Logoff. Input to these fields is handled by the macro, Update.FW2, defined below.
2. Days Since. This field is calculated by the formula behind the column header. It does not require any input from you.

Dialing a RBBS

The following user-defined macro, TCM Dailer.FW2, automates the dialup process. To use it, position the cursor on the row that corresponds to the name of the service you want to access and press **Alt**-< key > (you choose the key) to begin the dial up. From that point, the operation is under program control. Just sit back and wait to be connected. If your .TCM file includes a logon macro, it will begin to execute as soon as the file has been loaded from disk. Create an empty word frame. Press **EDIT FORMULA (F2)**, **ZOOM (F9)**, and enter the following formula,

---

```
; TCM Dialer.FW2
;
;   by Jack Hirsch
;      December 1, 1986
;
; Loads a selected .TCM file and then begins a Telecomm
; session.
;
; Declare local variables.
@local(tcm, command),
@echo(#OFF),
; Set the abnormal-end flag to false.
@isabend(#FALSE),
```

```
; Navigate to the field that contains the (.TCM) file.
@performkeys("{Ctrl-L}GTCM File{Return}"),
; Assign the .TCM filename to the local variable "tcm."
tcm := @textselection,
; Test for blank field.
@if(@len(tcm) < 1,
    @list(
          @beep(2000, 20),
          @prompt("TCM File field empty...", 22),
          @return(0)
    )
),
; Integrate the variable tcm into the @performkeys command line
; that loads the specified .TCM file from disk.
command := "@performkeys(""&"{Ctrl-D}G"&tcm&".TCM{Return}""&")",
; Execute the command line.
@command,
; Test for no-find (file not found).
@if(@isabend = #TRUE,
     @list(
          @beep(2000, 20),
          @prompt("TCM File not found...", 30),
          @return(1)
     )
),
; Turn echo on.
@echo(#ON),
; Begin the telecommunications session.
@performkeys("{Ctrl-A}TSB")
```

Recording the Access Date and Time

Recording the access date and time is initiated by another user-defined macro, Update. As before, you can assign it any Alt-<key> you want.

To setup, Update, create an empty word frame and label it with the appropriate **Alt-<key>** within curly braces on both sides of the frame name. For example, if you want to assign **Alt-U** (for Update), name the frame {Alt-U}. Then, press **EDIT FORMULA (F2)**, **ZOOM (F9)**, and enter the following formula,

```
; Update.FW2
;
;   by Jack Hirsch
;      December 1, 1986
;
; Updates RBBS database after logging off a service with a
; date and time stamp.
;
; Navigate to Last Access.
@performkeys("{CTRL-L}gLast Access{Return}"),
; Enter date and time.
@performkeys("@today{Tab}"),
; Enter time as text into Logoff.
@performkeys("{Del}"& " " & "{Alt-F4}"{Return}{Rightarrow}"),
; Update Days Since.
@performkeys("{Del}{Ctrl-Pgup}{F5}"),
; Navigate back to Last Access and sort descending.
```

```
@performkeys("{Ctrl-2}{Leftarrow}{Ctrl-L}D"),
; Save Update to disk.
@performkeys("{Ctrl-Return}")
```

This macro uses a macro already furnished by Framework II, **Alt-F4** (time). If you have changed or deleted this macro assignment, modify Update accordingly.

Now, **MOVE (F7)** this frame into your Library's macros section.

After you have logged off a bulletin board, update the RBBS record with the current date and time. To do this, press the **Alt-** < key > combination that you assigned for Update. The program macro, together with the formulas you input before, executes and does the following,

1. Enters the current date and time into Last Access with the **@today** function.
2. Enters the current time as text into Logoff.
3. Recalculates the number of days since the last logon. For the service you just called, this is zero.
4. Sorts the Last Access field for all records.
5. Saves the Update database frame to disk.

Conclusion

As you interact with various bulletin boards, create or modify .TCM files to reflect the correct protocol and parameters as well as any logon macros you want to include for each service. You can as well add, delete, or substitute fields in the RBBS database frame to reflect other information you want to track.

Amortization Schedule Using dBASE TOOLS FOR C

by Richard Bunter

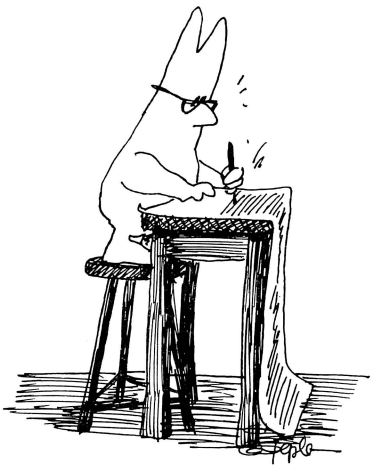
In addition to its interface to the C language, *dBASE Tools for C: The Programmer's Library* provides dBASE III PLUS users with ready-to-use financial, statistical, and mathematical functions. The financial functions include future value, internal rate of return, net present value, and amortization. With a few extra steps, you can easily incorporate these functions into a dBASE III PLUS program or application. The programs provided here, Amortize and Schedule, use the *dBASE Tools for C* to calculate and print an amortization schedule based on user input of these values,

1. Principal (initial amount of loan)
2. Rate of interest per year
3. Periods (that is, number of payments per year)
4. Payments made already
5. Years of loan (from start)

When you calculate the amortization of a loan, you are simply calculating the amount of the principal still outstanding after a given number of payments and at a given rate of interest. An amortization schedule reports the amount of each payment applied to principal and to interest over the life of the loan and the gradual death of the principal (that is, the running balance).

The first program, Amortize, sets up the return variables necessary for using *dBASE Tools for C* (the LOAD Cfunc and CALL Cfunc WITH command lines). Amortize then displays an input screen, READs the values, and calculates the balance and payments from the user input. Schedule then accepts the values from Amortize and displays or prints the amortization schedule as a custom report figuring the running balances and interest payments.

Note: Before running these programs, be sure to read your *dBASE Tools for C* documentation.



Getting Started with dBASE TOOLS FOR C

Before you load dBASE III PLUS or the Developer's Release, load *dBASE Tools for C* from DOS. From the drive or subdirectory where the Tools reside, type

```
DCT1
```

Now load dBASE III PLUS.

Note: If you are using a floppy disk system, use *dBASE Tools for C* from the B: drive. Amortize and Schedule should be copied to this drive as well. So, SET DEFAULT TO B if your *dBASE Tools for C* disk is in the B: drive and make sure that Amortize and Schedule are on the same disk before issuing the DO Amortize command.

If you are using a hard disk system, make sure that the *dBASE Tools for C*, Amortize, and Schedule are in your dBASE III PLUS subdirectory. After this, you are ready to amortize by typing,

```
DO Amortize
```

Amortize.PRG LOADs Cfunc, the *dBASE Tools for C* functions, and CALLs the necessary return variables. Then, the user input is accepted in accordance with the PICTURE clauses. The payments and amortization of the loan amount plus the interest are calculated using the PMT and AMORT functions from the *dBASE Tools for C* library. The current balance, the total loan amount (principal plus interest), and the regular payments are displayed. You are then given the option of displaying/printing an amortization schedule.

Schedule is called if the you choose to report. Schedule accepts the values from Amortize and uses LTRIM() and TRANSFORM() to format them for display and/or print. Then, the rest of the report is calculated. The interest portion of the regular payment is determined by multiplying the rate by the balance:

```
interest = balance * rate
```

The portion of the payment that applies to the principal is calculated by subtracting the interest payment from the regular payment:

```
principal = payment - interest
```

The running balance is calculated by subtracting the amount of each payment that is applied to the principal:

```
balance = balance - principal
```

The program ends with the creation of the report. If placed in a larger application, program control could return to another program or to a menu. Once the amortization schedule is displayed or printed, you can select other figures and run it again. Or, you could pass values from another program and calculate as many schedules as you have loans. **For further information** on using *dBASE Tools for C*, see the April 1986, June 1986, and August 1986 issues of ***TechNotes***.

Amortization by Example

As an example, we've supplied these sample values for an amortization schedule for a student loan: The initial value of the loan is $1800; the interest rate per year is 3%; the number of payments per year is 4, the number of payments made so far is 8; and the number of years from start is 3. Here's the schedule produced:

Amortization schedule, Page 1

Payments are $157.91

Initial Loan Balance was $1,800.00 at 3.00% per year

Balance at the start of this schedule is $619.22

Payment No.	Principal	Interest	Balance
9	153.27	4.64	465.95
10	154.42	3.49	311.53
11	155.58	2.34	155.96
12	156.74	1.17	-0.79

```
* Program.: Amortize.PRG
* Author..: Richard Bunter
* Date....: December 1, 1986
* Note(s).: Uses dBASE Tools For C to calculate the outstanding
*           balance of a loan and the regular payment per period.
*
IF TYPE("c_n_result") <> "N"
   CLEAR ALL
   LOAD Cfunc
   c_n_result = 0.00
   c_status   = 0
   c_error    = 0
   CALL Cfunc WITH "setdvar n"
   CALL Cfunc WITH c_n_result
   CALL Cfunc WITH "setdvar s"
   CALL Cfunc WITH c_status
   CALL Cfunc WITH "setdvar e"
   CALL Cfunc WITH c_error
ENDIF
SET TALK OFF
SET STATUS OFF
ON ESCAPE CANCEL
CLEAR
prin      = 0.00  && Real number. Initial amount of loan
yrate     = 0.00  && Real number. Rate of interest per year
yperiods  = 12    && Integer. Periods per year
payments  = 0     && Integer. Payments made so far
```

```
in_chains = 0     && Integer. Years to pay off the loan from start
print = .T.       && Logical. Print amortization schedule?
* ---Edit the default values.
@ 10,20 SAY "Initial amount of loan..... ";
        GET prin;
        PICTURE "@Z 99,999,999.99"
@ 11,20 SAY "Annual rate of interest..... ";
        GET yrate;
        PICTURE "@Z 99.99"
@ 12,20 SAY "Periods per year..... ";
        GET yperiods;
        PICTURE "99"
@ 13,20 SAY "Number of payments made so far ";
        GET payments;
        PICTURE "99,999"
@ 14,20 SAY "Number of years from start .... ";
        GET in_chains;
        PICTURE "@Z 99"
READ
rate    = (yrate/100) / yperiods
periods = yperiods * in_chains
* ---Calculate a regular payment over the life of the loan.
CALL Cfunc WITH "PMT " + STR(prin,,3) + "," + STR(rate,,3) + "," ;
                       + STR(periods) + "," + STR(payments)
payment = c_n_result
CALL Cfunc WITH "AMORT " + STR(prin,,3) + "," + STR(rate,,3) + ",";
                         + STR(periods) + "," + STR(payments)
STORE c_n_result TO balance, initbal
message = "The current balance is ";
          + LTRIM(TRANSFORM(balance, "##,###,###.##"))
@ 17,(80 - LEN(message)) / 2 SAY message
message = "You will be paying a total of ";
          + LTRIM(TRANSFORM(payment * periods, "##,###,###.##"))
@ 18,(80 - LEN(message)) / 2 SAY message
message = "In regular payments of ";
          + LTRIM(TRANSFORM(payment, "##,###,###.##"))
@ 19,(80 - LEN(message)) / 2 SAY message
@ 22,20 SAY "Do you want an amortization schedule? ";
        GET print;
        PICTURE "Y"
READ
@ 22,0
IF print
   DO Schedule
ENDIF
? "That's all, folks"
* EOP Amortize.PRG
```

```
* Program.: Schedule.PRG
* Author..: Richard Bunter
* Date....: December 1, 1986
* Note(s).: Called from Amortize to print out an amortization schedule.
*
* Variables used from calling program:
*
* prin    ::= Real number. Principal amount of loan.
* rate    ::= Real number. Rate of interest per period.
*         (example: 12% per year = .01 per month)
* periods ::= Integer. Number of periods at start of loan.
```

```
*               Must be greater than 0.
*               (example: 15 years = 180 monthly periods)
* yrate   ::= Real number. Yearly rate of interest.
* print   ::= Logical. To print or not to print
*         (that is the question)
*
print = .T.
@ 22,16 SAY "Send the amortization schedule to the printer? ";
        GET print;
        PICTURE "Y"
READ
IF print
   SET PRINT ON
ELSE
   CLEAR
ENDIF
* ---Maxlines can be initialized in the calling program also,
* ---Assumes STATUS is OFF.
maxlines = IIF(print, 60, 22)
page     = 1
pay_no   = payments + 1
DO WHILE pay_no <= periods
   ? "Amortization schedule, Page " + LTRIM(STR(page))
   ? "Payments are $" + LTRIM(TRANSFORM(payment ,"99,999,999.99"))
   ? "Initial Loan Balance was $" + LTRIM(TRANSFORM(prin,"99,999,999.99"))
   ?? " at " + LTRIM(TRANSFORM(yrate,"99.99")) + "% per year"
   ? "Balance at the start of this schedule is $";
      + LTRIM(TRANSFORM(initbal,"99,999,999.99"))
   ?
   ?
   ? "Payment No.        Principal          Interest          Balance"
   ? "-----------       -------------      ------------      -------------"
   line = IIF(print,  PROW(),  ROW())
    DO WHILE line < maxlines .AND. pay_no <= periods
       interest  = balance * rate
       principal = payment - interest
       ? STR(pay_no, 8)
       ?? SPACE(9) + TRANSFORM(principal, "9,999,999.99")
       ?? SPACE(8) + TRANSFORM(interest, "99,999.99")
       balance = balance - principal
       ?? SPACE(4) + TRANSFORM(balance, "99,999,999.99")
       pay_no = pay_no + 1
       line   = IIF(print, PROW(), line + 1)
    ENDDO
    page = page + 1
    IF print
       EJECT
    ELSE
       WAIT " Lightly depress just about any key to continue."
       CLEAR
    ENDIF
    ENDDO
    IF print
       SET PRINT OFF
    ENDIF
    RETURN
    * EOP Schedule.PRG
```

REPORT FORM Templates in dBASE III and dBASE III PLUS

by Peter N. Gold

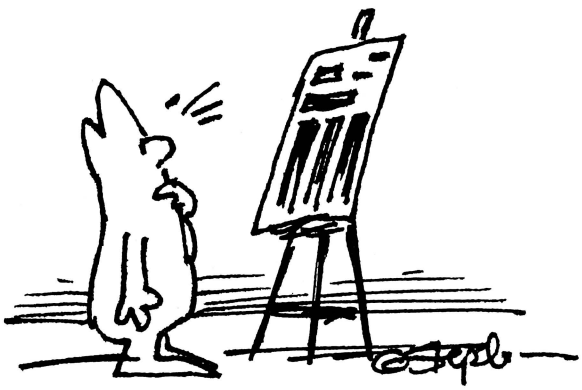
One frequent question at the Software Support Center is: "Can I use memory variables in the headings of my dBASE III PLUS REPORT FORMs? Yes, REPORT FORM allows a variable heading to be given to each use of the report with the HEADING option. For example,

REPORT FORM AcctRec HEADING "Interim Receivables for March"

This report would have the character string "Interim Receivables for March" on the top of each page along with the page number and date. If the report was used for final receivables, you can run the report with a HEADING to indicate this. The main body of the REPORT FORM heading, however, cannot be changed without actually MODIFYing REPORT. If the report is being run under the control of a command file by untrained operators, this task cannot be given to them. There is a better way — Cloner.PRG.

Cloner.PRG is a command file that speeds up the process of COPYing REPORT FORM several times. Each new REPORT FORM is given a new and unique filename. Any desired changes you want can be made to the REPORT FORM itself — break on different grouping criteria, summary or detail, double space, different titles for certain columns, etc. Each REPORT FORM only takes 2K of disk storage, so the overhead of multiple REPORT FORMS is not great.

The process of COPYing and MODIFYing REPORTs is not difficult, but working on repetitious tasks from the dot prompt contradicts the one purpose of a computer, speeding up the rote work. To use Cloner, create it with MODIFY COMMAND or another ASCII text editor, then type DO Cloner and follow the prompts.



When you run Cloner, you are prompted to select a drive designator and then database directory for the selected drive is displayed. Another prompt then advises you to select one of the displayed database files. Another request for a drive designator is followed by the display of the REPORT FORM's on the selected drive. From this list, you select the REPORT FORM to COPY. Once the copying process is completed, you are placed in MODIFY REPORT so that you can change the copy. You can then change the main report heading to reflect the actual heading you want and you can as well change the grouping criteria, summary or detail, double spacing, or different column headings.

The program code is commented to help you follow its flow. Several tests validate the user's response. In particular, the FILE() function is used to test for the existence of a valid database file and .FRM file. The AT() function is used to test for the inclusion of a filename extension and the SUBSTR() function is then used to pluck the filename from the entry by beginning at the first character and stopping short of the "." by one position:

SUBSTR(< variable > , 1, AT(".", < variable >) - 1)

The legal drive letters can range from A through Z. If you have fewer drives, delete those not present on your computer from the test string.

There are some possible modifications. Cloner.PRG can be modified to allow several copying processes before the modifications are performed. Or, once a modification process is complete, the prompt for copying can be redisplayed and another copy and modification can be performed. As written, only further copies of the same file are allowed.

```
* Program ...: Cloner.PRG
* Author ....: Peter N. Gold
* Date .......: December 1, 1986
* Verion.....: Developer's Release or dBASE III PLUS
* Notes .....: Copies REPORT FORMs from an existing report.
*              dBASE III and dBASE III PLUS do not allow
*              variables in report block headings.
*
PUBLIC datafile, repofile, checking, checking2, checking3,;
       mdatadrive, mrepodrive, again, bar
STORE .T. TO checking, checking2, checking3
bar       = REPLICATE("=",79)
alldrives = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
CLEAR
SET TALK OFF
```

```
? bar
? SPACE(25) + "R E P O R T  C O P Y  U T I L I T Y"
? SPACE(15) + "C L O N E S  d B A S E  I I I  R E P O R T  F O R M S"
? bar
?
?
DO WHILE checking
   WAIT "Which drive is your database file on? (A B C etc) " TO mdatadrive
   * ---Delete from the alphabet any drives not on your computer.
   IF .NOT. UPPER(mdatadrive) $ alldrives
       LOOP
   ENDIF
   ?
   DIR &mdatadrive.:
   ?
   ACCEPT "Type the datafile name or <cr> to quit (8 characters maximum) --> ";
          TO datafile
   DO CASE
   CASE LEN(TRIM(datafile)) = 0
       SET TALK ON
       CLEAR ALL
       RETURN
   CASE LEN(TRIM(datafile)) > 8
       IF "." $ datafile
          datafile = SUBSTR(datafile, 1, AT(".",datafile) - 1)
       ELSE
          LOOP
       ENDIF
   ENDCASE
   IF .NOT. FILE("&mdatadrive.:&datafile..DBF")
       LOOP
   ENDIF
   checking = .F.
ENDDO
DO WHILE checking2
   ?
   WAIT "Which drive is your report file on? ( A B C etc) " TO mrepodrive
   ?
   DIR &mrepodrive.:*.FRM
   ? "These report files already exist "
   ?
   ACCEPT "Type the name of a report file to clone --> " TO repofile
   IF LEN(TRIM(repofile)) = 0
      SET TALK ON
      RETURN
   ELSE
      IF LEN(TRIM(repofile)) > 8
         repofile = SUBSTR(repofile, 1, AT(".", repofile) - 1)
      ENDIF
   ENDIF
   IF .NOT. FILE("&mrepodrive.:&repofile..FRM")
       LOOP
   ENDIF
   checking2 = .F.
ENDDO
DO WHILE checking3
   ?
   ACCEPT "What name to give your next report? (8 letters maximum). <cr> to;
          quit " TO NEWNAME
   IF LEN(TRIM(newname)) = 0
      checking3 = .F.
      LOOP
   ENDIF
```

```
   IF FILE("&mdatadrive.:&newname..FRM") .OR. ;
      FILE("&mrepodrive.:&newname..FRM")
      ? CHR(7) + "That report already exists. Press Return and pick another "
      DIR &mrepodrive.:*.FRM
      ?
      WAIT "These report files already exist "
      LOOP
   ENDIF
   ?
   ? "Please wait a moment while copying &mrepodrive.:&repofile..FRM to';
      +..&mrepodrive.:&newname..FRM"
   COPY FILE &mrepodrive.:&repofile..FRM TO &mrepodrive.:&newname..FRM
   ? CHR(7) + "COPY completed. "
   ?
   WAIT "Press Return to modify the heading and anything else in the report. "
   USE &mdatadrive.:&datafile
   MODIFY REPORT &mrepodrive.:&newname..FRM
   WAIT "Do you want to make another copy of &mrepodrive.:&repofile..FRM ? ";
        TO again
   IF UPPER(again) = "Y"
      DIR &mrepodrive.:*.FRM
      LOOP
   ELSE
      EXIT
   ENDIF
   USE
ENDDO
CLEAR
* EOP Cloner.PRG
```

USAGE TIPS

Framework II

by Kent Irwin

Telecomm Macro to Trap on a Word

If you use telecommunications macros to automate your on-line activities, then you have noticed the limitations of the **@t** function. **@t** causes Framework II to wait for a particular character before proceeding with a macro. For instance, **@t**: will wait until a colon is received before allowing the rest of the macro to execute. However, this is frustrating in circumstances where the trap character may occur more than once. Examine, for example, this segment of a telecomm session connected to CompuServe,

!GO ASHFORUM

One moment please...

CompuServe ASHFORUM

Welcome to Ashton-Tate Forum, V. 4A(70)

Last visit: 18-Jul-86 14:31:53

Forum messages: 13083 to 13463

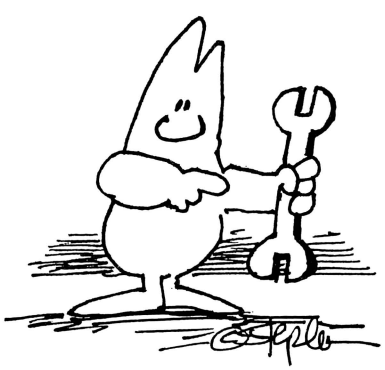
Last message you've read: 13450

No members are in conference.

Function:

The ! at the beginning is a prompt, at which you can type a CompuServe command. Suppose that you want your telecomm macro to issue the command GO ASHFORUM and then wait for the "Function:" prompt before you issue the next command. Using **@t**: would not work since there are five colons before the colon in the Function prompt.

The solution to this dilemma is to write a macro for the Library that can be called from the telecomm macro and that waits for a given string before proceeding. Here's an example,



```
; Word Trap
;
;   by Kent Irwin
;      December 1, 1986
;
@local(status, target),
status := #TRUE,
; This is the trap string.
target := "tion:",
@while(status,
      @if(@scan(target,
                telecommunications,
                @len(telecommunications) - 20
            ) <> #N/A!,
            status := #FALSE
      ),
      @performkeys("@w")
)
```

Install the Word Trap macro in your Library cabinet as {Alt-7}. Then, use it for the example session above with a telecomm macro like this,

GO ASHFORUM^M{Alt-7}B^M

which waits for "tion:" before issuing the next command ("B", for "read <B>ulletins"). The Word Trap macro scans the last 20 characters of the active telecomm frame, looking for the trap string. To change the trap string, just modify the assignment statement for the local variable "target."

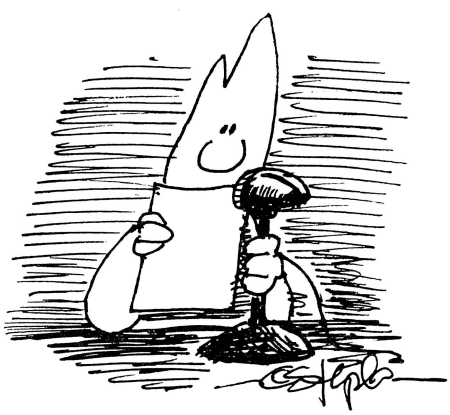
ON-LINE SERVICES

By Steve Silverwood

If you have a modem and a subscription to CompuServe or The Source, you can quickly access the same quality technical information that you have come to expect from ***TechNotes***—and you can do it through your computer system. To take advantage of this method of support, you must subscribe to one of the services listed below. There are no charges for using our on-line support services apart from the normal connect-time fees for each service. To get specific subscription information call one of the following numbers:

Service	Subscriber Information	Command to find Ashton-Tate
CompuServe	(800) 848-8199	GO ASHTON
The Source	(800) 336-3366	ATSIG

To give you a sample of this on-line support, ***TechNotes*** publishes monthly excerpts from the Ashton-Tate Forum on CompuServe. This material comes from messages our Forum users have posted on the bulletin board. All message headers have been removed to protect the identities of our users. Users' queries are printed in regular text. The responses of the Ashton-Tate technicians are printed in italics.



dBASE II, dBASE III, and dBASE III PLUS:

I have asterisks in a numeric field because of a REPLACE command that resulted in a few records being divided by 0. I can't figure out how to get dBASE III PLUS to recognize the asterisks so that I can REPLACE them with 0. I've tried all the functions I can think of. Thanks for your help.

If you have already done this and you want to recover from this situation, use the following algorithm substituting your values for Temp, the length, and number of asterisks in this program,

```
USE Test
REPLACE ALL Temp WITH 0 FOR STR(Temp, 5) = REPLICATE("*", 5)
```

If you are anticipating the kind of REPLACE statement where this might be a problem, use IIF( ) to prevent the divide by zero. Use the following general syntax for a REPLACE statement that prevents division by zero and places a zero into the field when the divisor is zero,

REPLACE ALL <field> WITH IIF(<divisor> = 0, 0 , <division expN>)

.....

I recently reformatted the hard disk in my Compaq portable with COMPAQ MS-DOS, version 3.1, and was suitably gratified with the smaller cluster size and resultant freed space. Since the reformatting, I have had difficulty whenever using an indexed file in dBASE II. The indexes get updated in RAM, but don't seem to be saved properly to disk. I realize that this is a vague explanation and that some other circumstance may be at work. I have attempted to isolate the problem by recopying my original dBASE II program files onto the disk without any success in fixing the problem. Any suggestions?

It appears that you are using dBASE II version 2.41 or below. Versions below 2.43 are not compatible with DOS versions later than 2.1x; so, you will need to update to dBASE II version 2.43* in order to use it in that environment. Contact Customer Service at (213)329-9989 to arrange for the update.*

.....

I need to set page length to 51 lines on my Epson FX-286 printer. It's set to 66 and nothing I do changes that setting. I've tried "? CHR(027/067/51) " but no luck. Can you suggest anything?

To initialize your printer for 51 lines with the codes you have given, use the following commands.

```
SET DEVICE TO PRINT
@  0,  0 SAY CHR(27) + CHR(67) + CHR(51)
SET DEVICE TO SCREEN
```

.....

Is there a way to handle large database files that cover more than one floppy aside without adding a hard disk?

dBASE III PLUS is not designed to handle database files spread over multiple floppies. About the best you can do is to process through the first database file on one disk, then prompt the operator to change disks, then continue processing using the second database file. The hitch to this is that your programs can't be sitting on the same disk as your data, since the disks will be swapped and the database files would have to be CLOSEd on the disk prior to changing it. If you use the RAM disk, your program files can reside on it and call up the data from the floppy. I'm working on a similar type of operation at home running in dBASE II on my Apple II + for my stamp collection with only 140K per diskette. I can't keep all of my stamp records on one disk; so, I have the dBASE II system running off drive A:, the programs running from B:, and thedisks with the data files being swapped in and out of drive C:. It's a little slow, but it works. When I get the whole thing working, I will probably upload it here.

.....

Can I set up a program to detect whether printer is ON or not when the SET DEVICE TO PRINT command comes into effect? If printer is not on, then the "Cancel, Ignore, Suspend?" error message is displayed and the screen becomes a mess.

You can intercept the "Printer not ready" error message with ON ERROR DO <program>. Then, in your error program, you can check the error code and take appropriate action.

I thought ON ERROR was only available with dBASE III PLUS using a LAN. I will try your suggestion. Thank you much.

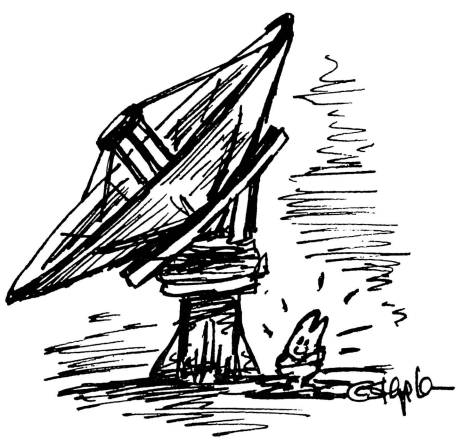
.....

I have a program file that keeps track of company name, phone number, etc., in my report files. I also have a field "Tst_d_dr" which holds the drive and path where the database file is located. I want to use this field in another program with a USE statement so that I can use a dual-floppy system or whatever the client wants. I have tried to say "USE Tst_d_dr + filename" but it doesn't work. I have also tried "SET PATH TO Tst_d_dr" and SET DEFAULT with no success.

You can use a macro to indicate the drive and path but only with a memory variable, not a field variable. You would add the lines,

```
STORE TRIM(Tst_d_dr) TO mdrive
USE &mdrive. <filename>
```

where <filename> is a literal. One caveat: Make sure when you STORE the drive and path that you add a backslash to the end of the string.



I am a beginner with dBASE III PLUS and in need of answers to a couple of questions. I would appreciate help from anyone.

1. Before printing a report, I need to dump a printer code to my printer to enable it to print in condensed mode. In BASIC, it would be "LPRINT CHR$(15);". Is there a way to do this in dBASE III PLUS?
2. When entering records, is there a ditto function? i.e., if one has to enter the same date on several records, is there a way to do this without re-entering the data for each record?

In answer to your first question, use the following commands,

```
SET PRINT ON
?? CHR(15)
SET PRINT OFF
```

or

```
SET DEVICE TO PRINT
@  0,   0 SAY CHR(15)
```

In answer to your second question, use the command SET CARRY ON to carry information from the previous record into the entry screen of the next record. To accept the default, hit **Return**, or you can type over the data there to change it in the new record.

.....

I want to SET FORMAT TO Screen1, then use DO Screen1 to display one record or use EDIT to display and allow EDITing of only one record. I also want to display a message on row 23, during EDITing only and not when displaying one. But when I issue EDIT it CLEARs the screen. Is there a way to display a message during the EDIT process without having to create two format files, one for EDITing and one for displaying?

If you are using dBASE III PLUS, look at the SET MESSAGE command. You can also DO <filename>.FMT and then put the message on line 23 before a READ. This doesn't work with multiple-page formats, though.

.....

There seems to be no documentation concerning the TEST() function in dBASE II version. 2.43*. Could someone tell me what exactly this function does, what it returns, and how it might be used? The help file entry for this function definitely whets my appetite!

According to the **Advanced Programmer's Guide**, *TEST() "determines whether the <expression> is valid, and if so, what type of data it contains." Doing a TEST() on the three types of data yield the results below:*

TEST(<character expression>)	*string length*
TEST(<logical expression>)	*1*
TEST(<numeric expression>)	*-6*

.....

Can you use dBASE III PLUS to network two workstations if you already have the network set up? It would be a Novell network. I have heard there's a network version of dBASE III PLUS. What is the difference between that and the regular dBASE III PLUS that I have now?

dBASE III PLUS, as shipped, is capable of operating on a network in either single or multi-user mode. In order to operate in multi-user mode you must have installed the dBASE Administrator which implements file and record locking. Instructions on how to do this are contained in both the Getting Started *manual and the* Networking with dBASE III PLUS *documentation, both of which are part of the dBASE III PLUS package.*

.....

Framework II:

Can somebody out there help me? I'm using Framework II to print on an NEC 3550 with the regular-cut sheet feeder (not the expandable one) and I keep getting a double formfeed every time! I've checked the switches on the NEC, but that's not the problem. Are there any bugs in the NEC driver I should know about? Does anyone have any suggestions? Any help would be appreciated.

You may be having this problem because you have Framework II set to print 66 lines per page. Ordinarily, this is not a problem, but with a sheet feeder you cannot print on every line of the page (there is generally a little margin at the bottom or top that you can't print on, because of the Out of Paper switch). Try printing a column of numbers from 1 to 66 and see how many lines fit on the page. Then, modify your **Print:Format:Lines per Page** *setting accordingly. I'm using a Toshiba P351 with only a single sheet feeder. I have used Setup to change my* **Print:Format:Lines per Page** *setting to 58 and to set Top and Bottom Margins to 0.*

.....

Framework II provides for conversion of fractions with single-digit numerators and denominators to the decimal equivalents when moving from word to spreadsheet frames. Is there a simple way to convert fractions with more than a single digit numerator and denominator?

Framework II fraction conversion, as used in **COPY (F8)** *and* **MOVE (F7)**, *is designed specifically for Dow Jones-type quotes that are no smaller than eighths. You can certainly write a macro that converts a fraction stored as text in a cell into a decimal number. The correct copying of the fraction from the word frame to the spreadsheet may require that the fraction be placed within quotes in the word frame before using* **COPY (F8)**.

.....

Is there any easy way to move a FRED program from a word frame to the formula area of another frame? Is there any way to copy a formula into a word frame or to send it through telecommunications? Going the other direction, is there a way to copy a formula into a word frame?

EXTEND SELECT (F6) *the formula,* **cut with Alt-F7**, *navigate to the destination and paste with* **Alt-F8**. These macros are provided in the Library.

MultiMate:

I am purchasing an Epson FX-286 printer. In the MultiMate User's Manual, the FX-286 is not identified as being supported. Is it supported and if so which .PAT file should be used? I downloaded the PATS-E.ARC file and the only new .PAT is the EPSONFX185.PAT. Any help would be greatly appreciated.

Yes, we support the Epson FX-286 printer. This printer has two modes, EPSON and IBM. To use the EPSON mode on this printer, set the DIP switches 1-4 to the ON position and use EPLQ1500.PAT. When you use EPLQ1500.PAT, you will get proportional spacing, but no line and box. To use the IBM mode on this printer, set the DIP switches 1-4 to the OFF position and use PROPRINT.PAT. When you use PROPRINT.PAT, you will not get printer proportional spacing, but you will get line and box. To get printer proportional spacing and line and box on the EPSON FX-286 printer, use EPFXLINE.PAT and the EPSON downloadable character set.

.....

MULTIMATE TECHNICAL SUPPORT

Submitted by Liz Burke

MultiMate Magic

Can I use MultiMate's header feature to place headers at the top of bound columns?

Yes, you can. To do this, you need two format lines. The first format line contains the tab stops, which centers the headings above the bound columns. The second format line defines these bound columns. Follow these steps to place headers at the top of bound columns.

1. On the first page of your document, press **F9** to position the cursor on the format line.
2. Enter tab stops that align with the bound columns in your document. Then, press **F9** to save the new format line.
3. Press **Alt-H** for the header symbol and then press **Return**.
4. Enter the column headers using **Tab** to move from one column to the next. When you are finished, press **Return**.
5. Press **Alt-H Return**.
6. To edit the second format line, press **Shift-F9**.
7. To define the bound columns, enter brackets in the format line, as described in the *MultiMate Reference Manual*, and then press **F9**.

Next, enter the columns of text. When you print the document, the headers print directly above each column.

How can I get indented footnotes in documents created with MultiMate Advantage?

Press **Alt-V** to bring up the footnote window, then press **F4** to indent.

How can I print several documents with consecutive page numbers?

To get consecutive page numbering across documents, use the automatic page numbering command (#) within the header or footer of each document. Before printing the documents, go to the "Print Parameters for Document" screen and enter the page number that you want to start with in the **Header/Footer First Page Number** option. Set this option to the appropriate number for each document.

Correcting Errors in Your Custom Dictionary

MultiMate's spell checker is designed to help you produce documents free of spelling errors and typos. MultiMate also lets you create your own dictionary of words that are not normally found in the *Merriam-Webster Dictionary*. The Custom Dictionary file (Clamfl.UH) is located on the MultiMate Speller/Dictionary disk.

If you ever accidently enter a misspelled word into your Custom Dictionary and misspell it in the same way in a document, MultiMate passes over it during spell checking. In addition, if the word is spelled correctly in the document, MultiMate flags it as incorrect.

If you have MultiMate Advantage version 3.60, you can correct entries made to your Custom Dictionary by using the Custom Dictionary Utility, located on the MultiMate Advanced Utilities disk. This utility allows you to view all of the entries in your Custom Dictionary so that you can delete and replace any misspelled words. To delete an entry from the dictionary, follow the steps below:

1. Execute the MultiMate Advanced Utilities program by typing,

   UTIL

   from the DOS prompt.

2. Use **Spacebar** to highlight the option **Custom Dictionary Utility** and then press **F10**.

3. From the "Custom Dictionary Utility" menu, highlight **Edit an Old File** and press **F10**.

4. The names of the Custom Dictionary files appear on the screen. Highlight **CLAMFL** and press **F10**.

5. From the "Edit Custom Dictionary" menu, select **Modify Custom Dictionary** and press **F10**.

6. Select **View/Delete Word** and press **F10**.

7. The first 48 words in your MultiMate Custom Dictionary display on the screen in alphabetical order. Press **PgDn** to scroll additional entries onto the screen. To delete a word, highlight it and press **Del**.

8. To save the changes, press **F10**.

9. Press **Return** to save the Custom Dictionary under the same name, Clamfl.UH. To return to DOS, press **Esc** four times from the "Modify Custom Dictionary" screen.

Print Custom Dictionary

Although MultiMate does not have a menu option for printing the contents of the Custom Dictionary, you can press **Shift-PrtSc** from the "View/Delete" screen to print out one screen of words at a time.

For more information on how to use the Custom Dictionary and its utility, refer to *MultiMate Advanced User's Guide*.

Creating Numbered Invoices in MultiMate Advantage

by Clay Carley

MultiMate, version 3.20 and later, has features that can be used to create numbered invoices and forms. This is useful if you send out invoices or forms to clients and want them consecutively numbered. For quick and easy keyboard entry, MultiMate can also pause for data entry at required places on the invoice or form. You can accomplish this by setting up a key procedure that creates the invoices from a template document. The template document contains the structure of the invoice and the commands to number the invoices consecutively.

The steps below demonstrate how to create a template document as well as how to set up a key procedure to create and number the invoices or forms. If you are not familiar with key procedures, refer to the *MultiMate Advantage Beginner's Guide* and the *MultiMate Advantage Reference Manual.*

Creating the Invoice Template

The invoice template contains the structure of the invoice form. Once the template is created, it is used by the key procedure to create numbered invoices. Incrementing the invoice number is done with MultiMate's vertical addition function, which requires a decimal tab. Follow the steps below to create an invoice template:

1. From the main menu, select option 2, **Create a New Document**.
2. Name the document Invtemp.
3. While in the document, press **F9** to change the format line.
4. Use **Spacebar** to move the cursor to column 60 and then press **Tab**, and **F9** to complete the change.
5. Next, press **Shift-F4** for a decimal tab and enter:

   0 **Return**

6. Press **Shift-F4** and enter:

   1 **Return**

7. Press **Shift-F4** again and enter a number, one less than the number with which you wish the invoice to begin. For example, if you want the invoice to begin with 100, enter 99.

   < number > **Return**

8. Now, design the invoice template you wish to use. If necessary, it can be a multipaged invoice. In addition, you can use MultiMate Advantage's line and box drawing feature to enhance the display of the invoice. For information on how to use this feature, refer to the *MultiMate Advantage Reference Manual.*
9. To save the invoice template, press **F10**.

Creating the Key Procedure

The key procedure uses the invoice template you just created (Invtemp) to generate the numbered invoices. To create this key procedure, follow the steps below:

1. To begin building a procedure, press **Ctrl-F5** from the main menu.
2. When prompted, name this procedure Proc.
3. Next, select option 1, **Edit an Old Document**.
4. Enter the name of the invoice template. In this example, it is Invtemp.
5. Press **F10** twice to move to the first page of the template document.
6. Press **Rightarrow** once and then press **Del**.
7. At the prompt "DELETE WHAT?" press **Alt-F5** to highlight the entire number.
8. Press **Del** again to complete the deletion.
9. Move the cursor to line three and press **Alt-F3**.

10. Move the cursor one character to the right and press **F8** to initiate a copy. The prompt "COPY WHAT?" appears.
11. Press **Alt-F5** to highlight the number.
12. Press **F8** again. The screen prompt "TO WHERE?" displays.
13. To complete the copy, move the cursor to the **Return** symbol on line 1 and press **F8**.

The next set of steps calculates a new invoice number and saves the template:

14. Move the cursor to line 3 and press **Alt-F3** to get to the decimal tab.
15. Press **Ctrl-F4** to do vertical addition.
16. Press **F10** to save the changes.
17. From the main menu, select option 2, **Create a New Document**.
18. Name the document Invoice, then press **Ctrl-F6 Return** (for procedure pause).
19. Press **F10** three times to move to the first page of the document.
20. Use **Spacebar** to move the cursor to column 60.
21. Press **Shift-F8** for External Copy.
22. Enter Invtemp for the document name.
23. Press **F10** to display the first page of Invtemp. The screen then displays the message "START COPY WHERE?"
24. From line three, press **Alt-F3** to move the cursor to the tab character. Then press **Rightarrow**.
25. Press **Shift-F8** to initiate the copy function. The screen prompt will say "COPY WHAT?"
26. Move the cursor to the end of the form. For a one-page document, press **End**. To complete the copy, press **Shift-F8**.

The final steps navigate the cursor to the appropriate places on the form or invoice and then pause the procedure for keyboard entry.

27. Move the cursor to the position on the screen where the first data item should be entered and press **Ctrl-F6**. **Ctrl-F6** instructs MultiMate to pause for keyboard entry.
28. Move the cursor to the next entry position and press **Ctrl-F6** again to insert a pause.
29. Repeat this process for each entry item.
30. To save this key procedure, press **Ctrl-F5**.

To execute this key procedure, press **Ctrl-F8** from the MultiMate main menu. Enter Proc when you are prompted for the key procedure name. Press **F10** to begin. When entering data into the invoice, press **Ctrl-F6** after each entry.

Although there are many steps to this key procedure, it does save time and keystrokes. The key procedure not only creates a new numbered invoice, it also navigates the cursor to each location where keyboard entry is required.

INTERNATIONAL

by Paul Bukauskas

A View of Javelin

As you might already know Ashton-Tate has an exclusive agreement to market, distribute, and support Javelin, a financial modeling package, outside the United States and Canada. Javelin is a business analysis and reporting program for the IBM PC and IBM compatibles. It offers up to ten ways to display data and four ways to enter it. It also offers a split-screen mode so you can see two views on the screen at once. A view is a perspective of data stored in a Javelin database. In addition to these features, Javelin also allows you to define custom time periods and to record frequently used keystrokes as macros. This article presents an overview of Javelin's unique method of entering, storing, and viewing data.

In Javelin, information is stored and retrieved from a central database, allowing you to share data among several worksheets and graphs. Sharing data in this way is not possible in conventional worksheets where data is anchored to a cell reference. In addition, formulas are defined in words instead of in row and column addresses. For example, you can say

```
Profit = Revenue - Cost
```

instead of

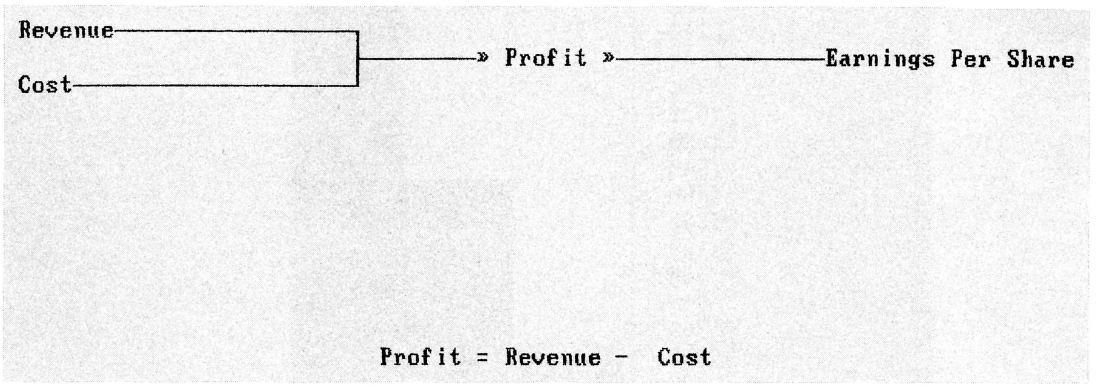
```
E12 = D12 - C12
```

Because data is stored in a central database, it is available to any one of Javelin's ten different views, each designed for a different business need. For instance, one view represents variables graphically, and another diagrams variables to show the relationships among them. Following is a brief description of each view.

Views

The Diagram View

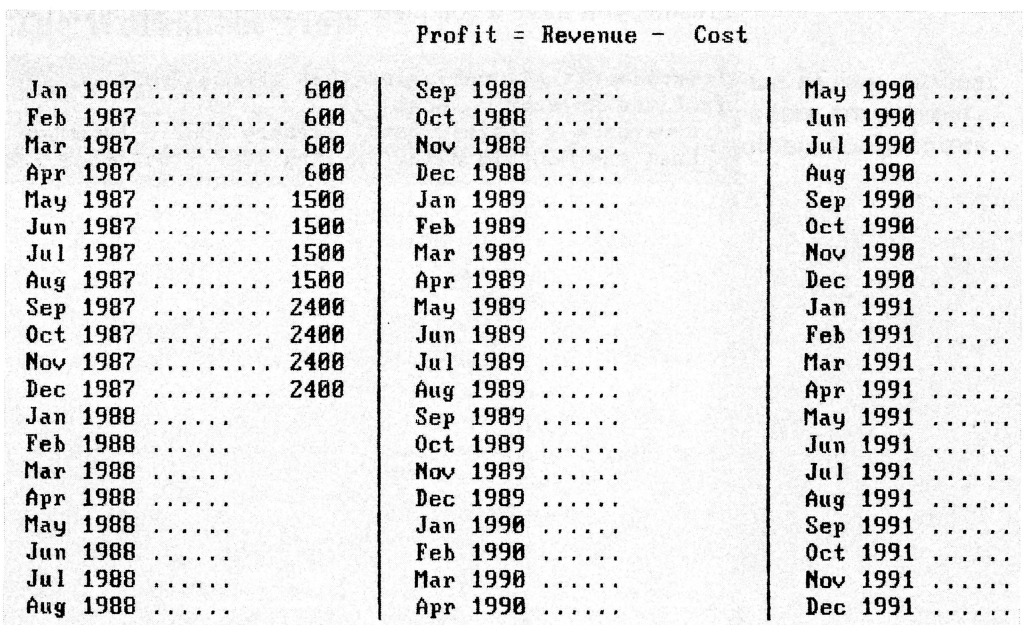
The Diagram View shows the relationship that one variable has to other variables. It not only shows which variables affect the first one, but also how the variable affects other variables. A diagram of Profit is shown below:



This diagram shows that Profit is affected by Revenue and Cost. It also shows how Profit affects Earnings Per Share.

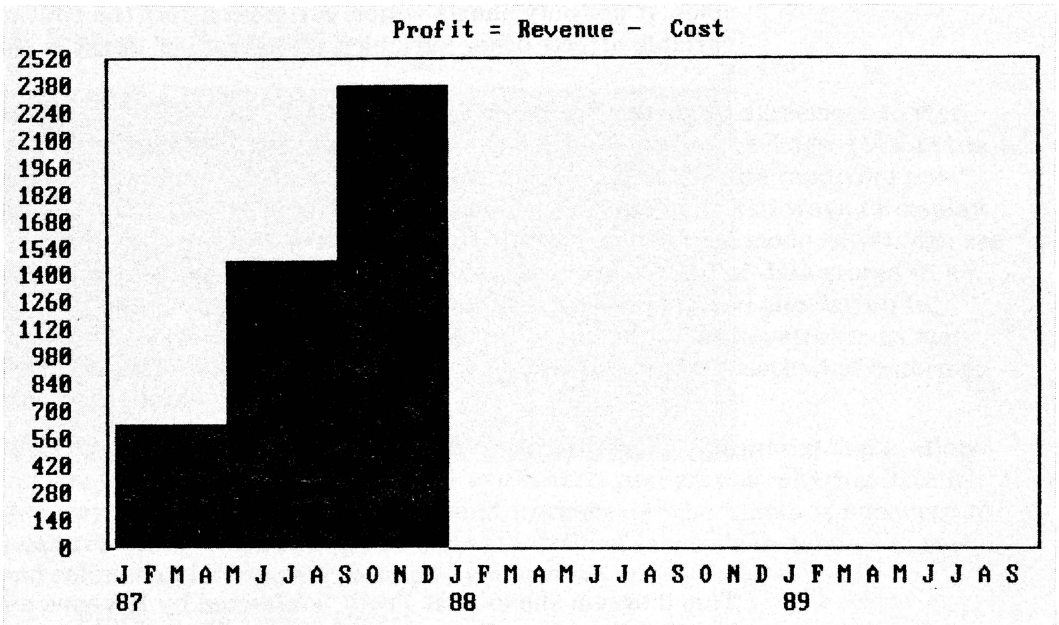
The Table View

The Table View presents data in a tabular format with dates to the left and values to the right. The dates displayed in the table default to a monthly division, though you can define any time period.



The Chart View

The Chart View represents one variable in bar-graph form. Like the Table View, it displays a single variable over a specific time period. The example below show a Chart View of Profits from January 1987 to December 1987.



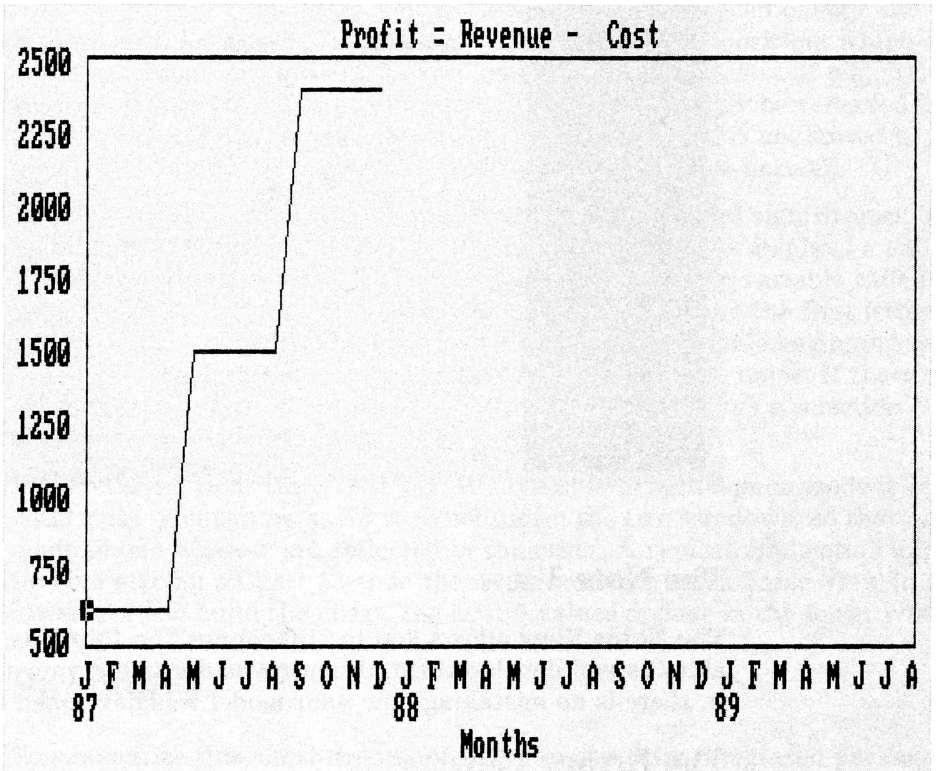
The Formula View

The Formulas View allows you to see some or all of the formulas defined in the central database. For instance, a Formulas View of Profit shows the definition of Profit, as well as the definitions of each variable that defines Profit. As a result, you have a complete description for each variable.

```
Formulas for all variables that affect Profit:
Profit = Revenue - Cost
· Revenue = ◀ Monthly data: January 1987 - December 1987 ▶
· Cost = ◀ Monthly data: January 1987 - December 1987 ▶
```

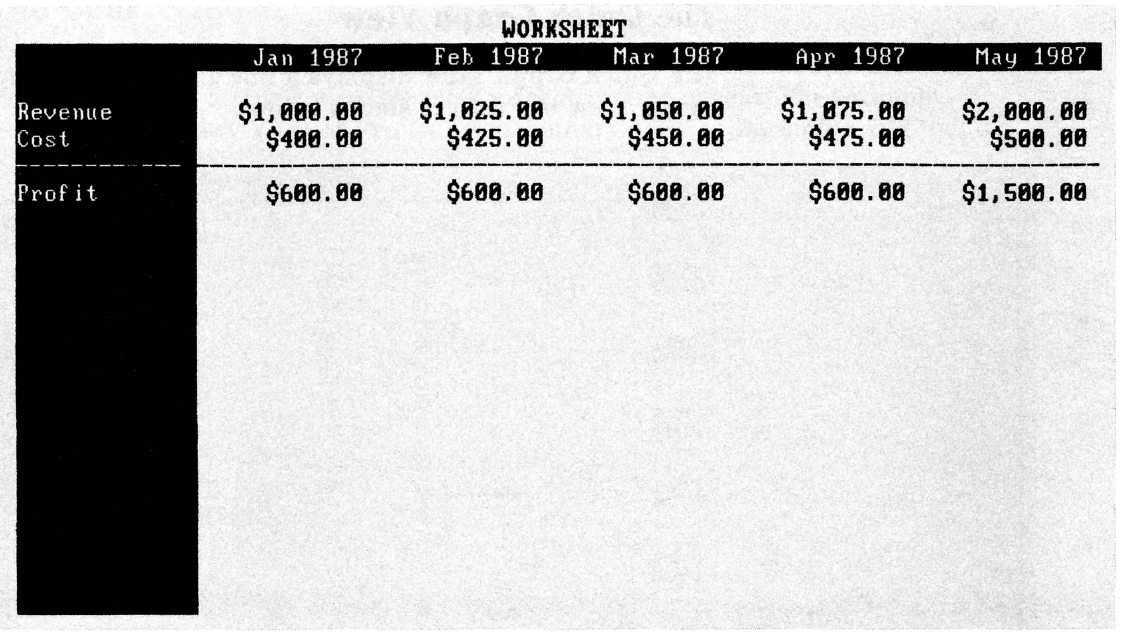
The Quick Graph View

The Quick Graph View displays a line graph for a single variable. A Quick Graph View of Profit is shown below.



The Worksheet View

The Worksheet View displays data in rows and columns just as conventional spreadsheets. The difference: once the row and column headers are entered, assuming these values were previously defined, Javelin automatically enters data into the worksheet.



WORKSHEET

	Jan 1987	Feb 1987	Mar 1987	Apr 1987	May 1987
Revenue	$1,000.00	$1,025.00	$1,050.00	$1,075.00	$2,000.00
Cost	$400.00	$425.00	$450.00	$475.00	$500.00
Profit	$600.00	$600.00	$600.00	$600.00	$1,500.00

The Note View

The Notes View allows you to "take notes" on formulas, constants, and variables as you develop them. Because you document your model as you develop it, there is no mistaking how your model was developed or conclusions drawn.

The Errors View

The Errors View provides a listing of any formula, constant, or variable that is either erroneous or undefined. Such errors occur when unlike time periods are mixed in the same formula or when constants are not defined.

The Macros View

The Macro View shows a listing of the keystrokes used to write a macro. Macros automate Javelin by allowing you to record frequently used keystrokes to "play" back when needed. Macros are created by programming in the Javelin macro language or by having Javelin record keystrokes from the keyboard.

The Graph View

The Graph View includes up to seven variables in a graph. There are seven graph types to choose from: line, bar, stacked bar, percent bar, horizontal bar, pie chart, and XY scatter plot. This differs from Chart View and Quick Graph View by offering more graph options and also by plotting more than one variable.

Data Entry

Of the ten views available, four allow you to add and update data in the central database. They are Table View, Quick Graph View, Worksheet View, and Chart View. For example, if you are in Chart View, you can change the values in the graph by moving the cursor to a higher or lower position, which subsequently changes the data in the central database. Regardless of which of the four views you select, any changes made will immediately be reflected in all other views. Again, this is because the data you change is not stored in the view, but in the central database from which the view is derived.

During data entry two keys on the keyboard have special significance: **Tab** and **F3**. **Tab** is referred to as the "SPELL" key because it displays a list of variables currently in use. If, for instance, you defined a variable called Working Capital, subsequent references can be made by typing the first letter of the word and then pressing the **Tab**. If this is the only variable beginning with the letter "W", Javelin automatically completes the variable name. If there are other variables that begin with "W", Javelin then displays a selection of variables for you to choose from.

F3 is referred to as the SYNC key. If Javelin is in split-screen mode when you enter data, you can press **F3** to synchronize the two windows, so that changes made to one window are reflected in the other. A typical application might involve placing a Chart View in the upper window and a Table View in the lower. As you build the chart, the actual values appear in the lower window.

Summary

To summarize this short discussion, Javelin is a fully integrated package that allows you to develop, present and fully document sophisticated business models. Because data is not anchored to a specific cell reference, it can easily be shared among and changed in different views. We have touched on only a few aspects of Javelin's features. Future articles will explore Javelin's powerful macro language as well as its ability to deal with time.

LETTERS

Indexing on Multiple Numeric Fields

Dear ***TechNotes:***

In response to F.C.'s letter (***TechNotes***, June 1986) on the subject of INDEXing a database file on two numeric fields, here is a way that doesn't involve conversion to character strings.

Consider a database file with two numeric fields (call them Primary and Secondary). We wish to INDEX this database file on the Primary field and within the Primary field by the Secondary field.

The dBASE III PLUS command is,

INDEX ON Primary + meps * Secondary TO <index file>

where "meps" is defined as a sufficiently small positive constant so that when multiplied by Secondary the product is smaller than the next incremental value of Primary. In essence, we are adding a trivial amount to Primary.

So, the key expression,

Primary + meps * Secondary

causes the order to be increasing by Primary and increasing by Secondary within Primary. The key expression,

Primary - meps * Secondary

causes the order to be increasing by Primary and decreasing by Secondary within Primary. The key expression,

-Primary + meps * Secondary

causes the order to be decreasing by Primary and increasing by Secondary within Primary. And the key expression,

INDEX ON - Primary - meps * Secondary

causes the order to be decreasing by Primary and decreasing by Secondary within Primary.

I hope that your readers will take advantage of this technique to INDEX multiple numeric fields in the future.

A.J.M.

Dear A.J.M.:
Thank you for sharing your indexing algorithm with our readers. I should note that while this will create an index that will sequence the records in the desired order, it will require some additional programming if a search through this index is desired. If the index file is created by the command,

INDEX ON Primary + meps * Secondary TO <index file>

then the command to locate the appropriate record in the index file would be,

SEEK mprimary + meps * msecondary

Plus Signs vs. Commas in Expressions

Dear ***TechNotes***:

I am trying to create a LABEL FORM in dBASE III PLUS that would print the City, State, and Zip fields on one line, such as:

John Doe
123 Elm Street
Smallville, KS. 12345

I used the expression

TRIM(City) + "," + State, ". " + Zip

but that line of the label printed an extra blank space between the State field and the period. What did I do wrong?

C.K.

Dear C.K.:

In the label form example you gave, the comma after the State field should have been a plus (" + ") sign.

When dBASE III PLUS encounters a comma in an expression, it separates the fields immediately before and after the comma with a blank space. That is what has happened here. Using the plus (" + ") character to concatenate the two items will eliminate the blank space.

Periods in Macro Expressions

Dear ***TechNotes***:

I was working with macros in a dBASE III PLUS program to access different files based on the contents of a variable and ran into some strange results.

When I use the command USE FD&mpfilenu.DBF, it doesn't pick up the file extension properly, but if I use the command USE FD&mpfilenu..DBF with two periods between the variable and the file extension, the command works fine. Why is this?

W.S.

Dear W.S.:
You do not need to specify the extension for a database file in the USE command unless it is other than ".DBF." Your problem, however, does point to the interesting behavior of the period when used with macro substitution. In this situation, it is used to terminate the expansion of the macro variable so that the text following the period will not be treated as part of the variable name. Since a file extension begins with a period, the first period must be included to separate the macro variable name from the rest of the string and then the second period is treated literal data.

Running on the IBM PC XT with the Enhanced Keyboard

Dear ***TechNotes:***

I want to install my copies of Framework II and dBASE III PLUS on a new IBM PC XT which has the IBM Enhanced Keyboard.

I understand from what has appeared in the press on this subject that there is an incompatibility between this new keyboard and my versions of Framework II and dBASE III PLUS.

What do I need to do to make these programs work with the new computer and keyboard?

P.J.R.

Dear P.J.R.:

Framework II will run on the new IBM XT and AT computers with the Enhanced Keyboard, provided that the appropriate keyboard driver is used. The default keyboard driver for Framework II version 1.1 supports both keyboards. If however, you have Framework II version 1.0, you will need a new keyboard driver. This driver is available upon request from the Ashton-Tate Software Support Center and may also be downloaded from our support forums on CompuServe and The Source.

To install the driver, run the Framework II Setup program and specify the driver file by name. When you're prompted to place the Setup disk into drive A: so that it can read the driver, insert this diskette instead, and the Setup program will read the driver file directly.

The new keyboard driver is designed to work with both the Regular and Enhanced Keyboards. If you have Framework II version 1.1 and the keyboard is not operating properly, install the keyboard driver, Oldibm.KB, included on the Setup disk.

dBASE III PLUS does not require any modification to run with the above configuration.

CORRIGENDA

1. In the April 1986 issue page 49, we erroneously stated that in dBASE III PLUS, READKEY() returns the wrong value if you terminate a full-screen operation with **Ctrl-W** or **Ctrl-End** and the contents of the full-screen edit are not changed. This is not anomaly. In dBASE III PLUS, READKEY() returns 270 if you terminate a full-screen operation with **Ctrl-W** or **Ctrl-End** whether or not the contents of the full-screen edit are changed. By contrast the Developer's Release, READKEY() returned 14 if there was no change and 270 there was a change.

   There is also a change in dBASE III PLUS to the value READKEY() returns when you press **Esc** to terminate a fill-screen operation. It now returns only the value 12.

   This is documented on page 41 of the November 1986 issue of ***TechNotes***.

2. On page 53 of the August 1986 issue of ***TechNotes***, there is an error in the second block of Framework II code so that the Backup program will not copy all of the selected files to the A: drive as stated in the text. To correct the problem, replace the line,

   ```
   Fwbackup := Fwbackup & "copy " & @setselection & "A:"
   ```

   with

   ```
   Fwbackup := Fwbackup & "copy " & @setselection & "A:
   "
   ```

3. On page 39 of the September 1986 issue of ***TechNotes***, there is an error in the second version of the Promptcenter user-defined function. The code,

   ```
   @if(@not(@iserr(@len(@item1))),
   @eraseprompt,
   @prompt(@item1,@max(0,(80-@len(@item1))/2) + 1))
   ```

   should be replaced with

   ```
   @if(@not(@iserr(@len(@item1))),
     @list(
       @eraseprompt,
       @prompt(@item1,@max(0,(80-@len(@item1))/2) + 1)
     )
   )
   ```

   Missing is an **@list** to group all of the subsequent formulas under the true path of the **@if** function.

4. On page 39 of the June 1986 issue of ***TechNotes***, the error on one the examples. The last indented example on the page that looks like,

   ```
   @performkeys("{Alt-K}")
   ```

   should be changed to

   ```
   @performkeys("{Alt-K}
   ```

TechNotes
Change of Address Form

If you have moved, or are planning to move soon, please let us know so we can provide you with uninterrupted service. To ensure that your subscription moves with you, please fill out this form and return it to:

TechNotes Subscription Service
P.O. Box 3729
Escondido, CA 92025-0929

Be sure to include your previous as well as your new mailing address (including zip codes).

PREVIOUS ADDRESS (or affix your CURRENT mailing label here)

Name_____

Company_____

Address_____

City _____ **State/Province**_____

Zip/Postal Code _____ **Country**_____

NEW ADDRESS

Name_____

Company_____

Address_____

City _____ **State/Province**_____

Zip/Postal Code _____ **Country**_____

NOTE: This change of address pertains to your *TechNotes* subscription ONLY. If you would also like to notify Ashton-Tate of your change in address (for upgrade and new product information), please write to: Ashton-Tate Customer Service, 20101 Hamilton Ave., Torrance, CA 90502.

COMING SOON: TECHNOTES MAC

We're proud to announce the publication of a newsletter designed to serve dBASE® Mac™ users: **TechNotes Mac. TechNotes Mac** is the monthly newsletter published by the Ashton-Tate Software Support Center. Each 16-page issue will be filled with the most accurate and most current technical information straight from our technicians.

SUBSCRIBE TODAY! Don't miss a single issue of the authoritative source for technical information about dBASE Mac! Use the convenient order form below to send in your subscription request to:

TechNotes Mac Subscription Service
PO Box 3729
Escondido, CA 92025-0929

For faster service, order **TechNotes Mac** by phone. Call:

(800)331-4164 (in California)
(800)828-2514 (Residents of other states)
(619)487-3212 (Residents of Canada)

Macintosh is a registered trademark and Mac is a trademark of Apple Computer Inc.

— —

Bill me: ☐

Amount enclosed:
☐ $35 (US)
☐ $42 (Canada and Mexico)
☐ $67 (Other)

Form of payment:
☐ Check (Make payable to Ashton-Tate in U.S. dollars drawn on U.S. bank.)
☐ VISA
☐ MasterCard

Card #_____

Expiration date _____ / _____ / _____

Signature_____

Name_____

Address_____

City_____ State/Province_____ Zip/PostalCode_____

Country_____ Phone number_____

PLEASE ALLOW 30 TO 60 DAYS FOR DELIVERY OF YOUR FIRST ISSUE.

1DM1

READER SUBMISSION FORM

If you have a programming tip, a handy formula, or a short utility, and you are interested in sharing it with other Ashton-Tate or MultiMate enthusiasts in a future issue of ***TechNotes***, please read the provisions below. After reading these guidelines, fill out this form and send it along with your submission(s) to the following address. Submissions must be made on disk (IBM or compatible format) in standard ASCII format or in Framework II (.FW2) format.

***TechNotes* Submissions**
Software Support Center
Ashton-Tate
20101 Hamilton Ave.
Torrance, CA 90502-1319

NAME _____ DATE _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

PHONE (During normal business hours) (_____) _____ - _____

_____

(SIGN HERE TO ACKNOWLEDGE YOUR UNDERSTANDING OF THE PROVISIONS BELOW)

To establish a clear understanding of the rights and usage of submissions, you are asked to first represent and warrant that:

1. You are the author of the material: that it is original and has not been heretofore published; that it does not infringe upon any copyright, proprietary right or any other right whatsoever; that it contains no matter that is contrary to law; and that you are the sole owner of the rights herein conveyed to us, and that such rights are free and clear.
2. We shall have the right to use your name, biography or other information pertinent to your submission.
3. We shall have the right, at our discretion, to edit the material to suit the requirements of our publication.
4. Suggestions, comments, and programming materials submitted to ***TechNotes*** are not for reply. Materials submitted for publication are considered contributions to the "pool" of information found in ***TechNotes*** for the advantage of our readers. Although, we do not pay or in any other way compensate you for your contribution, your name shall appear as author or contributor.

Thank you for reading ***TechNotes***. If you do not have a submission for us, write us anyway and let us know what you think about our publication. We certainly appreciate any feedback you may have.

SUBSCRIBE TODAY TO *TechNotes!*

Don't miss a single issue of Ashton-Tate's hottest source of technical know-how! Each monthly issue of ***TechNotes*** provides you with the most current, accurate technical information available from the Software Support Center on all Ashton-Tate and MultiMate products.

Call our toll free number to subscribe:

(800) 331-4164 (In California)
(800) 828-2514 (Residents of other states)
(619) 487-3212 (In Canada)

If you prefer, fill out the form below and return it with payment to:

***TechNotes* Subscription Service**
P.O. Box 3729
Escondido, CA 92025-0929

— — — — — — — — — — **(CUT HERE)** — — — — — — — — — —

WE ACCEPT CREDIT ORDERS! Call our toll free number for details. Make checks payable to ASHTON-TATE.

Amount Enclosed:

__ $50.00
__ $65.00 (Extra Postage Canada and Mexico)
__ $75.00 (Extra Postage other foreign countries)
__ Bill Me

Form of Payment: (check one)

__ Check (**U.S. dollars drawn on U.S. bank**)
__ VISA Number _____
__ MasterCard Number _____
Expiration Date _____
Signature _____

Check One: __ New Subscription __ Renewal

Name _____

Company _____

Address _____

City _____ **State/Province** _____

Zip/Postal Code _____ **Country** _____

Which product(s) do you use? __ **dBASE II** (D2)
__ **dBASE III** (D3)
__ **Framework** (FW)
__ **MultiMate** (MM)

Please allow 30 to 60 days for delivery of your first issue.

1000

TechNotes
BACK ISSUES ORDER FORM

Maybe your subscription to ***TechNotes*** just began and you want to have every issue. Maybe you don't subscribe, but want to receive last month's issue. Maybe you want to refer to an article in a past issue to help you design that special application.

Whatever the reason, order Back Issues! Add to your library of dBASE II, dBASE III PLUS , and Framework II technical reference material. To order your Back Issue(s), fill out this form, enclose your payment, and return to:

TechNotes Back Issues
Ashton-Tate Software Support Center
Attn: Kathy Smith
20101 Hamilton Ave.
Torrance, CA 90502-1319

Name _____

Company _____

Address _____

City _____ State/Province _____ Zip Code _____

Country _____ Phone Number (_____) _____-

All 1985 Back Issues are $10.00 each, when you order them separately. If you purchase all available 1985 issues as a set, the price is $50.00. All 1986 Back Issues are $7.00 each. All prices include postage and tax.

The following issues are currently available. (Those crossed out are out of print.) Check those you wish to order:

( ) FEBRUARY 1985
~~( ) MARCH 1985~~
~~( ) APRIL 1985~~
( ) MAY 1985
( ) JUNE 1985
( ) JULY 1985
~~( ) AUGUST 1985~~
~~( ) SEPTEMBER 1985~~
( ) OCTOBER 1985
( ) NOVEMBER 1985
( ) DECEMBER 1985 (includes 1985 Annual Disk and 1985 Cumulative Index)
( ) JANUARY 1986
( ) FEBRUARY 1986
( ) MARCH 1986
~~( ) APRIL 1986~~
( ) MAY 1986
( ) JUNE 1986
( ) JULY 1986
( ) AUGUST 1986
( ) SEPTEMBER 1986
( ) OCTOBER 1986
( ) NOVEMBER 1986

$_____ TOTAL AMOUNT ENCLOSED

( ) VISA ( ) MasterCard # _____

Expiration Date _____ Signature _____

ALL ORDERS MUST BE PREPAID. MAKE CHECKS PAYABLE TO ASHTON-TATE (IN U.S. DOLLARS DRAWN ON U.S. BANK). PLEASE ALLOW SIX (6) WEEKS FOR DELIVERY OF YOUR ORDER.

NOTE: Information contained in ***TechNotes*** is current as of the issue in which it appears. Therefore, some information found in the Back Issues may no longer be valid (such as Update Policies, etc.).

Cumulative Index

January 1986 through December 1986

Products:

D2	-	**dBASE II**
D3	-	**dBASE III**
DR	-	**dBASE III Developer's Release**
DP	-	**dBASE III PLUS**
DPA	-	**dBASE III PLUS ADMINISTRATOR**
DB	-	**dBASE Products**
FW	-	**Framework**
FW2	-	**Framework II**
MM	-	**Multimate**
UT	-	**Utilities**

Categories:

☆	-	**Articles**
¥	-	**International**
□	-	**Letters**
◇	-	**MultiMate**
○	-	**On-Line Services**
◀	-	**Sample Programs**
§	-	**Usage Tips**

Example: **product category**

Data Entry databases, FW2 ☆ Nov:31

Issue **Page**

@ row,col, DP ☆ Dec:15
@...CLEAR TO, DP ☆ Nov:15
@...GET, DP ☆ Jul:11, DP ☆ Sep:20, DP ☆ Dec:12
@...GET...PICTURE, DP ☆ Jul:12
@...GET...READ, DP1 ☆ Oct:12
@...SAY, DP § Apr:42, DP ☆ Jul:12, DP ☆ Sep:21
  to line 24, D2 § Jan:37
@...SAY...GET, DP ☆ Sep:21, DP ☆ Oct:13, DP § May:27
  forced choice, DP □ Jul:66
@...SAY...PICTURE, DP ☆ Sep:20
?/??, DP ☆ Oct:12

A

Abbreviations, FW2 ☆ Apr:36
ACCESS, DP ☆ Jun:6
Add.FW2, FW2 ◀ Nov:34,
  See also: Data entry
Age.PRG, DP ◀ Nov:9,
  See also: Date routines
ALIAS, DP § Jul:42
Amortization, DP ☆ Dec:36
Amortize.PRG, DP ☆ Dec:38
Amount.FW2, FW2 ◀ Nov:35,
  See also: Data entry
Ampm.PRG, DP ◀ Nov:7,
  See also: Time routines
APPEND, DP ☆ Oct:13
  on a network, DPA § Sep:35
APPEND FROM, DP § May:27, DP ☆ Oct:13, DP ☆ Oct:14
  DIF file,
  SYLK file, DP ☆ Nov:25
  WKS file, DP ☆ Nov:26
Arrays, DP ☆ Apr:9, DP ☆ Jun:25
  creating an array, DP ☆ Apr:12
  handling, UT ☆ Aug:38
  loading an array element, DP ☆ Apr:13
  removing an array, DP ☆ Apr:16
  retrieving the contents of an array, DP ☆ Apr:15
ASCII codes, DB □ Sep:63
ASCII conversion, MM ◇ Oct:56
Assembly language
  assembling, D2 ☆ Feb:16
  interface, D2 ☆ Feb:13
  programs, FW2 □ Mar:46
ASSIST, D3 § Mar:40
  CTOD(), DP § Apr:43
  LIST FOR, DP § Apr:43
  Position menu, DP ☆ Nov:15
Assumptions, FW2 ☆ Apr:38
AT(), DP □ Aug:66
Autoerr.PRG, DPA ☆ Aug:26
AUTOEXEC.BAT, DP ☆ Mar:29, UT ☆ May:20
Autolock.PRG, DPA ◀ Aug:25
AVERAGE, DP ☆ Oct:14, DP ☆ Nov:8

B

Backing up
  a hard disk, FW2 § Aug:53
  files, MM ◇ Jul:60
  frames, FW2 ◀ Sep:40
Batch files, DP ☆ Mar:29, UT ☆ May:19, DP ☆ Nov:19
  Ctrl-Z, UT ☆ May:20
  DOS ACCESS, FW2 □ Oct:63
Batch.FW2, FW2 ◀ Oct:63,
  See also: DOS Access
Bold.FW2, FW2 § Sep:37
Bookkeeping, DP □ Aug:65
Boxes, FW2 ☆ Apr:39
BROWSE, DP ☆ Feb:11, DP ☆ Aug:12, DP ☆ Dec:12
Bubsort.PRG, DP ◀ Jun:30
BUCKET in CONFIG.DB, DP ☆ Apr:20

C

C language, DP ☆ Apr:9, UT ☆ Aug:38
  sorting and searching, DP ☆ Jun:25
Calcsize.FW2, FW2 ◀ Jul:47
CALL, D2 ☆ Feb:15, DP ☆ Apr:9, DP ☆ Dec:12, DP ☆ Dec:26
Calling.FW2, FW2 ◀ Sep:30
CASE, DP § Apr:43
Case structure, FW2 § Jun:38
CATALOG, DP ☆ Dec:12
Catalog management, DP ☆ Apr:26
Cbinary.PRG, DP ◀ Jun:27,
  See also: Sorting and searching
Cell ranges
  copying, FW2 § Jun:40
Changing format lines, MM ◇ Oct:55
Character and numeric data hiding, DB □ Sep:63
Character functions, FW2 ☆ Dec:29
Character strings, D3 □ Mar:45, DP § Jun:33
Checker.PRG, DP ◀ Sep:7,
  See also: Mailmerge
Checkno.FW2, FW2 ◀ Nov:35
CHOOSE.COM, DP ◀ Mar:33,
  See also: Menus
@chr, FW2 § Aug:51
CHR(), D2 □ Apr:56
Cloner.PRG, DP ◀ Dec:42,
  See also: REPORT FORM
Close_req.PRG, DP ◀ Oct:32
CMONTH(), DP ☆ Nov:4
Color, DP § May:30, FW2 ☆ Jul:19, DP ☆ Mar:12
  EColor25, DP § May:27
  EColor43, DP § May:27
  SIGMA 400, DP ☆ Dec:16
Column capability, MM ◇ Apr:50
Column headers, MM ◇ Dec:52
COMMAND in CONFIG.DB, DP ☆ Apr:20, DP § Apr:44, DP □ Nov:61

Command lines
editing, DP ☆ Feb:7
prototyping, DP ☆ Feb:8
Commas.PRG, D3 ◀ Aug:36,
See also: Importing
Compare.PRG, DP ◀ Jun:28,
See also: Primitives
CompuServe
Ashton-Tate forum, UT ☆ Jun:11
CONFI256.DB, DP ☆ Apr:20
CONFI256.SYS, DP ☆ Apr:20
CONFIG.DB, DP § Apr:44, DP ☆ Jul:12, DP ☆ Oct:15, DP □ Nov:61, DP ☆ Dec:13
CONFIG.SYS, DP ☆ Apr:20, DP ☆ Dec:26
CONTINUE, DP ☆ Jul:13
COPY, D3 § Jan:38, DP § Oct:36
DELIMITED, DP § May:28, DP ☆ Sep:21, DP ☆ Oct:15
SDF, DP ☆ Oct:15
STRUCTURE EXTENDED, D2 ☆ Jan:10, DP § May:28, DP ☆ Jun:22, DP § Jul:42
Copy protection, UT ☆ Jul:28
Copying
text containing fractions, FW2 § Mar:42
Copyprog.FW2, FW2 ◀ Aug:53,
See also: Backing up
CP/M-80, D2 ☆ Feb:16
CP/M-86, D2 ☆ Feb:17
Cpdelim.PRG, DP ◀ May:28,
See also: COPY
CREATE, DP ☆ Sep:21, DP ☆ Oct:15
CREATE FROM, D2 ☆ Jan:10, DP ☆ Jun:22, DP § Jul:41, DP § Nov:37
CREATE SCREEN, DP § Jun:32, DP ☆ Dec:13
Ctrl-T, DP ☆ Nov:15
CREATE VIEW, DP ☆ May:18, DP ☆ Mar:7
Creeping text, MM ◇ Aug:59
Cseed2.PRG, D2 ◀ Jan:17
Cseed3.PRG, D3 ◀ Jan:19
CTOD(), DP ☆ Aug:12, DP ☆ Sep:22
Custom dictionary editing, MM ◇ Apr:51
Custom reports, D3 ☆ May:14, DP ☆ Jul:3, FW2 ☆ Sep:26
with format files, DP ☆ Jul:9
Customizing Framework II, FW2 ☆ Apr:34
C_binary.PRG, UT ◀ Aug:43,
See also: C language
C_compar.PRG, UT ◀ Aug:43,
See also: C language
C_convrt.PRG, UT ◀ Aug:44,
See also: C language
C_exchan.PRG, UT ◀ Aug:44,
See also: C language
C_insert.PRG, UT ◀ Aug:45,
See also: C language
C_partit.PRG, UT ◀ Aug:47,
See also: C language
C_select.PRG, UT ◀ Aug:45,
See also: C language
C_sequen.PRG, UT ◀ Aug:42,
See also: C language

D

Data,
See also: COPY
transferring one record between files, DB □ Oct:64
Data dictionary, DP ☆ Jun:22
Data encryption, D3 ☆ Feb:31
Data entry
databases, FW2 ☆ Nov:31
spreadsheets, FW2 ☆ Nov:31
under program control, FW2 ☆ Nov:32
Data tables, FW2 ☆ Feb:34
Data type conversion
numbers to text, FW2 ☆ Mar:25
text to numbers, FW2 ☆ Mar:25
Data validation
simulating the dBASE PICTURE clause, FW2 ☆ Oct:21
Database file size, DP § Nov:37
Database file structure, DP ☆ Mar:4
library, DP ☆ Jun:22
Database.FW2, FW2 ◀ Oct:23
Database.PRG, DP ◀ Apr:28
@date, FW2 § Aug:52
Date range
searching techniques, DP ☆ Aug:30
Date routines, FW2 ☆ Dec:29
calculating ages, DP ☆ Nov:9
calculating working days, DP ☆ Nov:5
date averaging, DP ☆ Nov:9
date formatting, DP ☆ Nov:4
general date formats, DP ☆ Nov:4
setting the system, DP ☆ Nov:3
Date strings
comparing English dates, FW2 § Aug:52
DATE(), DP ☆ Sep:3
Dateavg.PRG, DP ◀ Nov:9
Datefind.PRG, DP ☆ Aug:32
Datefunc.FW2, FW2 ◀ Sep:19,
See also: Exporting
Day.PRG, DP ◀ Nov:7
dBASE TOOLS FOR C, DP ☆ Apr:9,
See also: C language
macro functions, UT § Oct:37
supported compilers, UT § Oct:37
dBASE, running from Framework, FW2 ☆ Mar:27
DBF(), DP ☆ Apr:22
DBRUN.EXE, DR § Jan:40
Debit and credit signs, DP □ Aug:65
DEBUG, DP ☆ May:24, DP ☆ Mar:29
Decrypt.PRG, D3 ◀ Feb:32
Decryption, D3 ☆ Feb:32, DB □ Sep:63
DELETE, D3 § Jan:39
Deleting spaces, FW2 § Jul:49
Delimited files, MM ◇ Sep:54
Dictionaries
custom, MM ◇ Dec:52
personal, FW2 § Mar:42
DIF files, FW2 ☆ Sep:12, DP ☆ Nov:23
DISPLAY
HISTORY, DP ☆ Feb:7
OFF, DP ☆ Jul:13
TO PRINT, DP ☆ Oct:16, DP ☆ Dec:14
Display adapters and drivers, FW2 ☆ Jul:22
Display.PRG, DP ◀ Apr:30
DO, DP ☆ Aug:13
DO WHILE, DP § Apr:44, DP ☆ Sep:22, DP ☆ Oct:16
DOS
3.1 subdirectories, D2 § Mar:39
SET PATH, FW2 ☆ Apr:34
text files, FW2 □ Mar:46
DOS Access, FW2 § Jun:41
running batch files, FW2 □ Oct:62
Downloading, FW2 ☆ May:4, FW2 ☆ Feb:23
Draw a box macro, FW2 ◀ Apr:39
DTOC(), DP ☆ Sep:22, DP ☆ Nov:10

E

EDIT, DP ☆ Feb:11, DP ☆ Jul:13, DPA ☆ Aug:25
Editing, FW2 ☆ Jan:15
Elapsed time, FW2 § Jul:46
Elapsed_time.FW2, FW2 § Jul:46
Encrypt.PRG, D3 ◀ Feb:31
Encrypted database files, DP ☆ Aug:24
Encryption, D3 ☆ Feb:31, DB □ Sep:63
Engineering notation, FW2 § Oct:38
Engldate.FW2, FW2 § Aug:52
Engsci.FW2, FW2 ◀ Oct:39
Err.PRG, DP ◀ Apr:32, DP ◀ Oct:27
Error messages
by number, DP § Nov:37
"Document Full", MM ◇ Oct:54
"Too many files open", FW2 § Aug:52
Error trapping, FW2 ☆ Feb:39
Eval2.PRG, D3 ◀ Jan:22
Eval3.PRG, D3 ◀ Jan:23
Exchange.PRG, DP ◀ Jun:29,
See also: Primitives
Exponentiation, D2 ☆ Jan:18, DP ☆ Sep:22
Exporting,
See also: Importing
Lotus 1-2-3 release 2 files, DP § May:32
VisiCalc DIF file, FW2 ☆ Sep:12
Extended memory requirements, FW2 § Mar:42
External copy, MM ◇ Jul:59

F

Fdate.PRG, DP ◀ Nov:4,
See also: Date routines
Fields limitation, DP □ Jul:67
FIELDS(), DP ☆ Apr:23
File handles, DP ☆ Dec:26
File header, DP § Nov:37
File structures
transferring, DP § Jul:41
FILE(), DP ☆ Oct:17

Filenames, DP § Jun:32, DP § Jul:41
FillChar.FW2, FW2 ◀ Dec:31,
See also: Date routines
FillDate.FW2, FW2 ◀ Dec:29,
See also: Date routines
FIND, DP ☆ Oct:17, DP ☆ Nov:16
Foreign character sets, FW2 ¥ Aug:63
Format files
for custom reports, DP ☆ Jul:9
Formatting problems, MM ◇ Aug:60
Forms view, DP ☆ Feb:11
aligning numbers in a database, FW2 § Nov:43
Formula edit
displaying a frame while editing, FW2 § Sep:38
FOUND(), DP ☆ Nov:16
Frame labels
style attributes, FW2 § Sep:37
Frames
backing up, FW2 § Sep:40
Frames menu, FW2 ☆ Feb:22
Freearry.PRG, DP ◀ Apr:16,
See also: Primitives
Function keys
programming, D3 □ Jun:48
Fw_2_dif.FW2, FW2 ◀ Sep:16,
See also: Exporting

G

GETS in CONFIG.DB, DP ☆ Apr:20
GOTO, DP ☆ Aug:13
Go_recno.FW2, FW2 ◀ Oct:40
Graph menu, FW2 ☆ Feb:22
Graphics, FW2 ☆ Jul:17
Grouped report, DP ☆ Jul:3,
See also: Reporting
headings, DP ☆ Jul:4

H

Handles.ASM, DP ☆ Dec:27
Hard disk, FW2 ☆ Mar:27
HELP, DP ☆ Aug:19, DP ☆ Dec:13
Help system, DP ☆ Aug:19
Hex-dec.CMD/PRG, D2 ◀ Feb:18,
See also: Assembly language
Hex.FW2, FW2 ◀ Apr:37
Hexadecimal equivalents, FW2 ☆ Apr:37
Higher sorts
bubble sort, DP ☆ Jun:29
HISTORY, DP ☆ Feb:8, D3 § Mar:40, DP ☆ Aug:13
HISTORY in CONFIG.DB, DP ☆ Apr:20

I

@if, FW2 § Jun:38
IIF(), DP □ Apr:57, D3 ☆ May:15, DP ☆ Aug:14
@...SAY, DP ☆ Mar:11
INDEX, DP ☆ Mar:10, DP □ Jul:67
REPORT FORM, DP ☆ Mar:8
Importing
converting PFS files, FW2 ☆ Mar:36
Framework 1.x files, FW2 § Mar:41
Lotus 1-2-3 release 2 files, DP § May:32
Symphony to Framework II, FW2 ☆ Jul:24
transferring data from Lotus, D3 □ Feb:44
with no delimiters, D3 ☆ Aug:35
INDEX, D3 □ Apr:56, DP ☆ Aug:14, DP ☆ Oct:17
conditional index keys, DP ☆ Mar:10
DTOC(), DP ☆ Nov:16
"File is not accessible", DP ☆ Nov:16
numeric fields, DP □ Dec:62
on a network, DPA § Sep:37
SET CENTURY ON, DP ☆ Nov:15
Index key, DP □ Jul:67
Indexing
by week, D3 □ Oct:62
character field with descending numeric, D3 □ Jun:48
conditional, DP □ Apr:57
creating files in subdirectory, D3 □ Apr:56
date and time fields, DP □ May:41
"Record is not in index", D3 □ Jan:47
steet addresses, DP § Nov:40
INKEY(), DP § Aug:50, DP □ Nov:61
INPUT, DP § Apr:44
@inputline, FW2 □ Oct:62, FW2 ☆ Nov:33
INSERT, DP ☆ Dec:15
Insert a page, MM ◇ Aug:62
Insertion sorts, UT ☆ Aug:45
INSTALL, DP ☆ Jun:5
Interactive mode, DP § Apr:44
International
ASCII 161, UT ¥ Oct:61
DBPRINT.PTB, UT ¥ Oct:59
index files, UT ¥ Sep:62
language versions, UT ¥ Sep:61
MODIFY COMMAND, UT ¥ Sep:62
spell checker, UT ¥ Oct:61
Invoices, MM ◇ Dec:53
@isabend, FW2 ☆ Jan:34, FW2 ☆ Feb:38, FW2 § Nov:43
Italic.FW2, FW2 § Sep:37

J

Javelin, UT ¥ Dec:56
JOIN, DP ☆ May:17, DP § Jun:33

K

Key expression, DP § Aug:48
Key filtering, FW2 § Oct:38
Keyboard locking up, MM ◇ Aug:60
@keyfilter, FW2 ☆ Mar:18
Keyset.FW2, FW2 ◀ Oct:38

L

Label
automated mailing labels, FW2 ☆ Mar:14
LABEL FORM, DP ☆ Jul:13, DP ☆ Oct:18,
See also: Reporting
reporting, DP ☆ Aug:27
using two database files, D3 □ Mar:45
Left-justifying, D2 § Mar:40
Library
abbreviations, FW2 ☆ Feb:25, FW2 ☆ Apr:34
assumptions, FW2 ☆ Apr:34
macros, FW2 ☆ Feb:24, FW2 ☆ Apr:34
macros for programmers, FW2 ☆ Feb:26
printing templates, Fw2 ☆ Apr:34
user-defined functions, FW2 ☆ Apr:34
Library attachment, MM ◇ Aug:62
Line and box drawing, MM ◇ Aug:62
Line feeds, D2 □ Apr:56
Line spacing
variable within a single frame, FW2 § Oct:41
LIST, DP ☆ Oct:16, DP ☆ Dec:14
vertical form, DP ☆ Apr:22
LIST HISTORY
to file, DP ☆ Feb:8
List.PRG, DP ◀ May:25,
See also: Printing with dBASE
Listform.PRG, DP ◀ Apr:23,
See also: LIST
Listhist.PRG, DP ◀ Feb:8
LOAD, D2 ☆ Feb:14, DP ☆ Apr:9, DP § Jun:34, DP ☆ Aug:14, DP ☆ Dec:26
Loadarry, DP ◀ Apr:14,
See also: Primitives
LOCATE, DP ☆ Jul:13
Locate menu, FW2 ☆ Feb:21, FW2 § Nov:43
Locking, DPA ☆ Aug:25
Logarithms, D2 ☆ Jan:18
Logical fields to character strings, DP ☆ Mar:9
Logon macros, FW2 ☆ Jun:16
LString.FW2, FW2 ◀ Jan:25

M

Maclib.FW, FW ☆ Feb:29
Macros, FW2 ☆ Apr:35, DP □ Dec:63,
See also: Library
{**Alt-B**}, FW2 ◀ Apr:39
{**Alt-L**}, FW2 ◀ Jan:30
{**Alt-M**}, FW2 ◀ Jan:31
{**Alt-R**}, FW2 ◀ Jan:31
{**Alt-W**}, FW2 ◀ Jan:29
Mailing labels, FW2 ☆ Mar:14
Mailmerge
labels, FW2 § Mar:43
programming, DP ☆ Sep:3
Main.FW2, FW2 ◀ Sep:17,
See also: Exporting
Main.PRG, DP ◀ Apr:27, DP ◀ Aug:31, DP ◀ Sep:6
Makearry, DP ◀ Apr:12,
See also: Primitives
Medit.BAT, DP ◀ Nov:21,
See also: MultiMate Advantage
Memo file
reducing size of, DP ☆ Sep:31

Memory resident
editors, DP ☆ Feb:9
programs, FW2 □ Feb:45
Memory variables, DP § May:30
Memoslim.PRG, DP ◀ Sep:32
Memotrim.PRG, DP ◀ Sep:34
MENU.BAT, DP ◀ Mar:32,
See also: Menus
MENU.DAT, DP ◀ Mar:33,
See also: Menus
Menus
color, DP ☆ Mar:12
Column menu, DP § Jun:35
Locate menu, DP § Jun:35
navigation menu, DP § Jun:35
Options menu, DP § Jun:35
PC/MS-DOS, DP ☆ Mar:29
with batch files, DP ☆ Mar:29
Merge, DP ◀ Sep:9,
See also: Mailmerge
Metaprimitives, UT ☆ Aug:44
Microsoft Mouse, FW2 ☆ Dec:7
@mid, FW2 ☆ Jan:24
MIN(), DP ☆ Dec:15
MODIFY COMMAND, DP ☆ Aug:14
MODIFY REPORT, DP ☆ Aug:15, DP ☆ Oct:18
MODIFY SCREEN
Ctrl-T, DP ☆ Nov:15
MODIFY STRUCTURE
an alternative to, D2 ☆ Jan:10
Modstruc.PRG, D2 ◀ Jan:11
MString.FW2, FW2 ◀ Jan:28
MS_mouse.FW2, FW2 ◀ Dec:9
MultiMate
Communications board, MM ◇ May:36
converting document content architecture, MM ◇ Nov:57
converting info-handling file to ASCII, MM ◇ Nov:56
converting Lotus worksheet to .DOC file, MM ◇ Nov:59
creating and sorting info-handling files, MM ◇ Sep:58
drive letters, MM ◇ Nov:55
GraphLink, MM ◇ Apr:51, MM ◇ May:36
Just Write, MM ◇ May:38
Keyboard, MM ◇ May:36
LAN versions, MM ◇ May:38
On-File, MM ◇ May:37, MM ◇ Oct:56
sort and print info-handler datafiles, MM ◇ Sep:58
supported products, MM ◇ Apr:52
system requirements, MM ◇ Apr:51
Technical support, MM ◇ Apr:52
trouble shooting, MM ◇ Apr:51
MultiMate Advantage, MM ◇ Apr:50
using with dBASE III PLUS, DP ☆ Nov:18
Multirpt.PRG, DP ◀ Aug:28,
See also: Reporting
MVARSIZ in CONFIG.DB, DP ☆ Apr:20

N

Naming ranges, FW2 § Jul:45
Network administrator
BACKUP and RESTORE, DPA § Jul:43
editing a record, DPA ☆ Aug:25
"File is in use by another", DPA § Jul:42, DPA § Sep:35
GOTO/SKIP, DPA § Jun:36
installation problems, DPA § Jul:43
LIST, DPA § Jul:44
LOCATE, DPA § Oct:36
MODIFY STRUCTURE, DPA § Jul:44
"Network is Busy", DPA § Jun:36
nondedicated server, DPA § Sep:36
PROTECT, DPA § Jun:37
read-only files, DPA § Jun:37
record locking, DPA ☆ Aug:25
RUN, DPA § Jun:37
SET DELETED ON, DPA § Jul:45
SET PRINTER TO, DPA § Jun:38
Network techniques
APPENDing to a database file, DP ☆ Oct:27
avoiding file locking, DP ☆ Oct:26
database file access, DP ☆ Oct:28
Newpage.PRG, DP ◀ Apr:25,
See also: LIST
Nextline.FW2, FW2 ☆ Apr:41
Normal.FW2, FW2 § Sep:37
Novell network
disk configuration, DP ☆ Jun:4
error messages, DP ☆ Jun:6
installing and uninstalling, DP ☆ Jun:5

O

On-Line, UT ○ Jun:12
Optim.PRG, DP ◀ Oct:29
Option.PRG, DP ◀ Apr:29
Opt_err.PRG, DP ◀ Oct:33
Opt_file.PRG, DP ◀ Oct:31
Opt_main.PRG, DP ◀ Oct:30
Opt_proc.PRG, DP ◀ Oct:32
Opt_req.PRG, DP ◀ Oct:33
Opt_wait.PRG, DP ◀ Oct:32

P

Pagetop.PRG, DP ◀ Jul:8
@panel1, FW2 ☆ Jan:14, FW2 ☆ Jan:34
@panel2, FW2 ☆ Jan:14
PARAMETERS, DP ☆ Jul:14, DP ☆ Aug:15, DP ☆ Sep:22
Partitioning a set of data, UT ☆ Aug:46
Passing parameters, D2 ☆ Feb:16
Password, DP ☆ Aug:23
Patchprt.PRG, DP ☆ Dec:23,
See also: Printing with dBASE
Payee.FW2, FW2 ◀ Nov:35
PCOL(), DP ☆ Sep:23
PEEK and POKE, D2 ☆ Feb:13
@performkeys, FW2 ☆ Jan:34, FW2 ☆ Feb:21, FW2 ☆ Feb:39, FW2 § Jun:39
pfs:file transfer utility, FW2 ◀ Mar:37,
See also: Importing
Picture.FW2, FW2 ◀ Oct:24
Primitives, DP ☆ Jun:27, DP ☆ Apr:12, UT ☆ Aug:38
compare operation, DP ☆ Jun:28
exchange operation, DP ☆ Jun:29
@print, FW2 ☆ Nov:27
Printing with dBASE
Control codes, DP ☆ Dec:17
Ctrl-C, DP ☆ Oct:16
from a program, DP ☆ May:25
LOAD and CALL, DP ☆ May:25
program listings, D3 ☆ Mar:21
Printing with Framework II
print menu, FW2 ☆ Feb:23
printer control codes, FW2 ☆ Aug:4
printer drivers, FW2 ☆ Aug:3
printer problems, FW2 ☆ Aug:10
supported printers, FW2 ☆ Aug:6
Printing with MultiMate
Diablo 630 underlining, MM ◇ Jul:59
document in queue, MM ◇ Aug:59
HP LaserJet, MM ◇ Jun:44, MM ◇ Jul:61
HP ThinkJet (Model 2225C), MM ◇ Jun:43
justified type, MM ◇ Jul:59
line spacing, MM ◇ Aug:60, MM ◇ Nov:55
merge print stopping, MM ◇ Jul:60
page length limits, MM ◇ Jul:60
pitch changes, MM ◇ Jul:61
print error accessing, MM ◇ Aug:61
"Printer Needs Attention", MM ◇ Aug:61
proportional spacing, MM ◇ Jun:44, MM ◇ Jul:61
removing underlines, MM ◇ Jul:60
selective merging, MM ◇ Nov:55, MM ◇ Aug:62
stop print, MM ◇ Aug:62
superscripts and subscripts, MM ◇ Jul:59
Tandy 1000 printing, MM □ Jul:62
typewriter mode, MM ◇ Apr:50
Print_report.FW2, FW2 ☆ Nov:30
PROCEDURE, DP ☆ Oct:15
Prompt messages, FW2 § Sep:39
Promptcenter.FW2, FW2 § Sep:39
Protect, DP ☆ Aug:23
Pull-down menu options, FW2 ☆ Jan:15

R

Random access memory (RAM)
installed memory, DP ☆ Apr:20
Random numbers generation, D2,D3 ☆ Jan:17
Random2.PRG, D2 ◀ Jan:18

Random3.PRG, D3 ◀ Jan:20
RANGE, DP ☆ Aug:16, DP ☆ Dec:12
READ, DR § Jan:40, DP ☆ Aug:16
READKEY(), DP § Nov:41
Reassigning keyboard characters, FW2 ¥ Aug:63
RECNO(), DP ☆ Oct:18
Record locking, DPA ☆ Aug:25
Record number
simulating GOTO, FW2 § Oct:40
REINDEX, DP ☆ Nov:16
"File is not accessible", DP ☆ Nov:16
Relating database files, DP § Aug:48
RELEASE ALL, DP ☆ Oct:19
Remove.PRG, DP ◀ Jun:34, *See also: Character strings*
REPLACE, DP □ Jul:68, DP ☆ Aug:16, DP □ Aug:67, DP ☆ Jul:14, DP § Jul:42
Replace.FW2, FW2 ◀ Nov:44
REPORT FORM, DP § Jun:34, DP ☆ Jul:14, DP ☆ Sep:23, DP ☆ Oct:19, DP ☆ Dec:41
calculated fields, D3 ☆ May:15
formatting numeric and logical fields, D3 ☆ May:15
printing on multiple lines, D3 ☆ May:16
programming, DP ☆ Jul:3
reports from related databases, D3 ☆ May:17
right-justifying char fields, D3 □ May:40
suppressing zeros, DB □ Sep:66
totals with character output, DP □ Aug:67
Reporting, DP ☆ Jul:9
custom grouped, DP ☆ Jul:3
invoices, MM ◇ Dec:53
paginated column reports, DP ☆ Aug:27
with LABEL FORM, DP ☆ Aug:27
@rept, FW2 ☆ Jan:34
RESTORE, DP § May:30
RETURN TO MASTER, DP ☆ Jul:14
Right-justifying, D3 □ May:40, FW2 § Jul:49, DP ☆ Dec:6
RLOCK(), DPA § Sep:37
Row and column values in word frames
simulating @...SAY, FW2 ☆ Sep:26
RString.FW2, FW2 ◀ Jan:26
@run, FW2 ☆ Mar:27, FW2 § Jun:41, FW2 □ Oct:62
RUN, D3 □ Apr:56, DPA § Sep:36
Rundbase, FW2 ◀ Mar:27, FW2 § Jun:41

S

Sales.PRG, DP ◀ Jul:7
SAVE, DP ☆ Dec:15
Savestru.PRG, DP ☆ Jun:23, *See also: Database file structure*
Saving bytes when copying, MM ◇ Oct:55
Say.FW2, FW2 ◀ Sep:27
Say_Init.FW2, FW2 ◀ Sep:29
Say_Main.FW2, FW2 ◀ Sep:27
@scan, FW2 ☆ Jan:26
Schedule.PRG, DP ◀ Dec:39
Screen drivers, FW2 § Jan:42, FW2 ☆ Jul:16, FW2 ☆ Jul:23
Screen dumps, DP ☆ May:24
Scroll1.PRG, DP ◀ Nov:12, *See also: Date routines*
Scroll2.PRG, DP ◀ Nov:13, *See also: Date routines*
Security utility, DP ☆ Aug:23
SED, D2,D3 □ Mar:44
Seekarry.PRG, DP ◀ Apr:15, *See also: Primitives*
@select, FW2 § Jun:38
Semicolon, DP § Apr:44, DP § Nov:41
Sending line feeds, D2 □ Apr:56
@sense, FW2 ☆ Nov:27
SET, D3 □ Jan:49
ALTERNATE, DP ☆ Sep:25
CALL TO, D2 ☆ Feb:15
CARRY, DP ☆ Oct:19
CATALOG, DP ☆ Apr:26, DP ☆ Dec:15, DP § Nov:41
COLOR, DP § May:30, DP ☆ Mar:12, DP § Nov:41
DATE, DP ☆ Nov:10
DATE AMERICAN, DP ☆ Nov:10
DATE ANSI, DP ☆ Nov:10
DATE FRENCH, DP § Apr:45
DEBUG ON, DP ☆ Oct:19
DELIMITERS, DP ☆ Oct:19
DEVICE, DP ☆ Dec:15
FIELDS, DP ☆ Mar:4, DP ☆ May:18, DP § Oct:36, DP ☆ Mar:7, DP § Apr:45
FILTER TO, DP ☆ Mar:7, DB May:11, DP □ Nov:62
FORMAT TO, DP ☆ Dec:16
FUNCTION, DP ☆ Sep:25
HISTORY TO, DP ☆ Aug:16
MARGIN TO, DP ☆ Sep:23
ODOMETER TO, DP § Jun:35
ORDER TO, DR § Jan:40, DP § Apr:45, DP ☆ Aug:17
PATH TO, D3 § Jan:40, D3 □ Apr:56
PRINT ON, DP ☆ Nov:17
PRINTER TO LPT1, DPA § Sep:63
RELATION TO, DP ☆ Mar:4, D3 □ Mar:45, DB May:11, DP ☆ May:17, DP § May:31, DP ☆ Jul:14, DP □ Jul:68, DP § Aug:48, D3 □ Sep:64
SCOREBOARD, DP § Apr:45
STATUS, DP § Apr:45
TALK ON, DP ☆ Aug:17
VIEW TO, DP ☆ Mar:5, DP § Apr:46
@setdirectory, FW2 § Jun:41, FW2 □ Oct:62
@setdrive, FW2 § Jun:41
@setformula, FW2 ☆ Nov:27
@setmacro, FW2 ☆ Nov:36
Setprint.ASM, DP ◀ Dec:21, *See also: Printing with dBASE*
@setselection, FW2 ☆ Feb:21
SETUP, FW2 ☆ Feb:29
Sharing data
DIF, SYLK, WKS files, DP ☆ Nov:23
with MultiMate Advantage, MM ◇ Sep:54
Sheet feeder, FW2 § Aug:54
Sheet.FW2, FW2 ◀ Oct:23
Sidekick
running with Framework II, FW2 □ Jul:68
Single-user security, DP ☆ Aug:24
SORT, DP ☆ Jul:15, DP ☆ Aug:17, DP ☆ Sep:25, DP ☆ Oct:20
Sorting and searching, DP ☆ Jun:25, UT ☆ Aug:38
Spacing.FW2, FW2 ◀ Oct:43
Spell checker macro, FW2 § Jul:48
Spool device, DPA § Sep:35
Spreadsheet labels
word wrapping, FW2 § Nov:44
Spreadsheets
calculating overall width, FW2 § Jul:47
Startday.PRG, DP ◀ Oct:27
STATUS in CONFIG.DB, DP ☆ Apr:20
STORE, DP ☆ Jul:15, DP ☆ Oct:20
String functions
user-defined functions/macros, FW2 ☆ Jan:25
STUFF(), DP □ Aug:66, DP ☆ Dec:3
Subscripts and superscripts, D3 § Mar:41, FW2 § Aug:51
SUBSTR(), DP § Nov:42
@sum, FW2 § Mar:42
SUM, DP ☆ Oct:14
SUPERLoK, UT ☆ Jul:28
BACKUP and RESTORE, UT ☆ Jul:31
error messages and numbers, UT ☆ Jul:35
installation, UT ☆ Jul:32
key disk, UT ☆ Jul:32
supported hardware environment, UT ☆ Jul:28
trouble shooting tips, UT ☆ Jul:29
Support services
data libraries, UT ☆ Jun:13
bulletins, UT ☆ Jun:14
SYLK file, DP ☆ Nov:25

T

@t, FW2 § Dec:45
Tab stops
variable, FW2 ☆ Mar:18
Table view, DP ☆ Feb:11
Tabs
removing, FW2 ☆ Jan:34
TabStrip.FW2, FW2 ◀ Jan:35
TCM Dialer.FW2, FW2 ◀ Dec:33, *See also: Telecommunications*
.TCM file definitions, FW2 ☆ Nov:27
TCM_Builder.FW2, FW2 ◀ Nov:28
Telecommunications
Ashton-Tate forum, UT ☆ Jun:11
automatic logon macros, FW2 ☆ Jun:16
configuration panel, FW2 ☆ Apr:8
convert .PAR files to .TCM files, FW2 § Mar:43
downloading, FW2 ☆ May:4
format panel, FW2 ☆ Apr:7
hardware setup, FW2 ☆ Apr:3

library macros, FW2 ☆ Jun:20, FW2 § Dec:45
menu, FW2 ☆ Dec:32
point & dial, FW2 ☆ Apr:6, FW2 § Jul:48
serial port #, FW2 ☆ Apr:8
.TCM frames, FW2 ☆ May:3
uploading, FW2 ☆ May:6
@textselection, FW2 ☆ Jan:34
Thesaurus, MM ◇ Apr:50
@time, FW2 § Jan:42
Time routines
averaging dates, DP ☆ Nov:8
displaying days, DP ☆ Nov:8
setting the system, DP ☆ Nov:3
twelve-hour time format, DP ☆ Nov:7
TIME(), D3 ☆ Jan:19, DP ☆ Sep:3
TimeFrame
running with Framework II, FW2 □ Jul:68
TOTAL, DP ☆ Nov:17
@trace, FW2 § Jul:50
Tracing, FW2 § Jul:50
Trans date.FW2, FW2 ◀ Nov:34
TRANSFORM(), DP ☆ May:16, DP ☆ Nov:17
Translating data, FW2 ☆ Jul:24
TYPE, DP ☆ Aug:18
TYPEAHEAD in CONFIG.DB, DP ☆ Apr:20

U

Underline.FW2, FW2 § Sep:37
UNINSTAL, DP ☆ Jun:5
UPDATE, DP § May:31
Update.FW2, FW2 ☆ Dec:34, *See also: Telecommunications*
Uploading, FW2 ☆ May:4
USE, DP § Nov:42, DP ☆ Dec:16
User-defined functions, FW2 ☆ Apr:36
UTIL, MM ◇ Oct:56

V

View.PRG, DP ◀ Feb:12, *See also: Forms view*
Views, DP ☆ Mar:4
Virtual database file, DP ☆ Mar:7

W

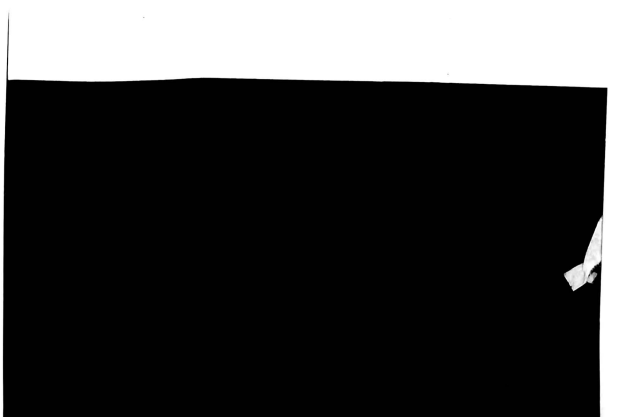
Weekday1.PRG, DP ◀ Nov:6, *See also: Date routines*
Weekday2.PRG, DP ◀ Nov:6, *See also: Date routines*
Winners, D3 ◀ Jan:20
WKS file, DP ☆ Nov:26
Word check macro, FW2 § Jul:48
Word frames
range search and replace, FW2 § Nov:43
Word processors, DP ☆ Nov:18
Wordwrap.PRG, DP ◀ Sep:10, *See also: Mailmerge*
Wrap Trap.FW2, FW2 § Dec:46, *See also: Telecommunications*
Wrap.PRG, DP ◀ Apr:24, *See also: LIST*
Wrap_cell.FW2, FW2 ◀ Nov:46
@writetextfile, FW2 § Jun:40

Z

ZAP, DP ☆ Dec:16
ZOOM (F9), FW2 § Sep:38

Notes

Notes



TechNotes
P.O. Box 3729
Escondido, CA 92025-0929

ADDRESS CORRECTION REQUESTED

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 27
CULVER CITY, CA